Proceedings of the 2007 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY 20-22 June 2007

A Family of Efficient Key Predistribution Schemes for Pairwise Authentication

Mahalingam Ramkumar

Abstract— **The growing need for ad hoc establishment of security associations (SA) in networks that may include resource constrained nodes has sparkled renewed interest in key predistribution schemes (KPS). KPSs which employ only inexpensive symmetric cryptographic primitives also support ID-based distribution of secrets to eliminate the need for dissemination of certificates or other public values. KPSs are however susceptible to collusions. The complexity associated with deployment and use of any KPS that can restrict collusions of very large number of nodes is influenced by several factors like bandwidth overheads, computational complexity, storage requirements, overheads for distributing secrets etc. We introduce a class of simple scalable KPSs with several compelling advantages over other KPSs in the literature.**

I. Introduction

A key distribution scheme (KDS), typically consisting of a key distribution center (KDC) and N nodes, is a mechanism for distributing secrets to all nodes in the network to facilitate establishment of "cryptographic bonds," or security associations (SA) between the nodes. While many types of SAs like one-to-one, one-to-many, and group SAs are possible, in this paper we limit ourselves to one-to-one SAs which facilitate mutual authentication through pairwise shared secrets.

While schemes based on the symmetric Needham-Schroeder protocol [1] (like Kerberos) demand very low resource requirements for participating nodes, the need for the ongoing involvement of a trusted server renders such approaches unsuitable for many application scenarios. Perhaps the most common approach in scenarios where ad hoc authentication is necessary is the use of public key schemes in conjunction with one or more certificate authorities. However the computational overheads required for asymmetric cryptographic primitives and the bandwidth overheads for exchanging public key certificates (or certificate chains [4]) can render such approaches unsuitable in many application scenarios.

More recently, several ID-based public key schemes or ID-based encryption (IBE) schemes [2], [3] have also received substantial attention. For such schemes the unique ID assigned to any entity simultaneously serves as the certified public key. ID-based approaches facilitate entities to choose meaningful IDs, like a descriptive string that reflects an entities real-life identity). Alternately, every node may be assigned a 160-bit or 128-bit ID, which is the cryptographic hash of the descriptive string. For instance, Alice described by a string $\bar{A}$ = "Alice B. Cryptographer, AnyTown USA," can be assigned an ID which is the SHA hash of the string $\bar{A}$. Such schemes require a key escrow who *assigns* private keys (corresponding to the chosen public key) to every entity. Thus unlike certificates based schemes where the private keys are chosen by the entities, for identity based approaches the private keys are *assigned* to each entity by a KDC. While IBE schemes eliminate the overheads needed for dissemination of certificates, they are in general computationally more expensive than certificates based public key schemes.

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

A. ID-based Key Predistribution Schemes

Key predistribution schemes (KPS) cater for ID-based approaches for key distribution, and employ only symmetric cryptographic primitives. A KPS consists of a key distribution center (KDC) and N nodes with unique IDs. The KDC chooses a set of P secrets $\mathbb{S}$. Each node is provided with a set of k secrets. The set of k secrets provided to a node A (a node with ID A), viz., $\mathbb{S}_A$, is a function of $\mathbb{S}$ (the secrets chosen by the KDC) and the ID A of the node. Or

$$\mathbb{S}_A = \mathcal{F}(\mathbb{S}, A). \tag{1}$$

Any two nodes A and B with secrets $\mathbb{S}_A$ and $\mathbb{S}_B$ respectively, can independently discover a security association (SA) K_{AB} as

$$K_{AB} = \mathcal{G}_A(\mathbb{S}_a, B) = \mathcal{G}_B(\mathbb{S}_b, A) \tag{2}$$

where $\mathbb{S}_a \subset \mathbb{S}_A$ and $\mathbb{S}_b \subset \mathbb{S}_B$. In other words, node A employs a subset of $m =\mid \mathbb{S}_a \mid$ of the $k =\mid \mathbb{S}_A \mid$ secrets assigned to A and the public ID of B to evaluate the shared secret K_{AB}. Node B on the other hand, employs a subset of its secrets and the public ID of node A to evaluate K_{AB}. For some KPSs $m = k$. In other words all k secrets of a node A need to be used by A to evaluate an SA with any node. There also exists KPSs for which $m << k$.

The total number of nodes N (or the maximum network size) that can be assigned secrets is only limited by the number of bits used for representing the ID of each node, as each node requires a unique ID. As in IBE schemes, nodes may also choose meaningful strings as IDs (or cryptographic hashes of such strings). Unfortunately KPSs are susceptible to collusions of nodes. An attacker who has exposed all secrets from many nodes may be able to compromise SAs between nodes that have not been physically compromised. An n-secure KPS can resist a coalition of up to n nodes pooling their secrets together.

For most n-secure KPSs $k \propto n$, and $P \propto n^2$. For some KPSs the number of secrets that need to be used to evaluate the SA, m, may be smaller than k, while for some $m = k$. Obviously one can improve the collusion resistance of any KPS by choosing a "large enough" $k \propto n$. For example, if a KPS can resist collusions of say several tens or even hundreds of thousands of nodes, their susceptibility to collusions may not be a practical concern. Unfortunately, improving collusion resistance of KPSs is accompanied by an increase in resource requirements - both for the nodes and the KDC.

The resource requirements of a KPS has several components. While the resource requirements for the KDC may not be a severely mitigating factor, for very large networks this would also require consideration. As every node may approach the KDC to obtain their secrets, we would require that the KDC be able to compute the secrets required for any node (by executing function $\mathcal{F}(A)$ for node A) in a reasonable amount of time, with reasonable demands on storage and computational complexity. The resource requirements for each node to compute pairwise secrets - for example for nodes A and B to execute $\mathcal{G}(\mathbb{S}_A, B) = \mathcal{G}(\mathbb{S}_B, A) = K_{AB}$) - however deserves more consideration, especially in scenarios involving resource constrained nodes. The resources required for this purpose could take many forms like storage complexity, computational complexity, memory complexity, number of secrets used for evaluating $\mathcal{G}()$ etc. Thus the overall cost associated with any KPS could be a weighted sum of several components.

While the specific weights may change with changes in technology, in general, for application scenarios involving resource constrained nodes, storage is perhaps the least expensive of resources. For example, for battery operated devices (as most resource constrained devices are expected to fall under this category) the need to prolong battery life will call for the use of low power processors. However, increasing the storage capabilities of such devices (for example with flash memory) may not have much effect on the battery life. Storage is the resource with the fastest Moore's law growth rate, especially for mobile devices. Thus, efficient KPSs should strive to take advantage of cheaper resources to reduce their dependence on more expensive resources. This is the motivation for a class of novel KPSs presented in this paper.

In Section 2 of this paper we provide an overview of several KPSs. In Section 3 we present the new KPS, multiple basic key distribution (MBK) and discuss several advantages of MBK over existing KPSs, especially for scenarios requiring a very high level of collusion resistance (or large n). We highlight why existing KPSs are not well suited for facilitating large n. In Section 4 we introduce a variant of MBK, hashed MBK (HMBK) and compare performance of the two schemes. Conclusions are offered in Section 5.

II. Key Predistribution Schemes

KPSs can be broadly classified into non scalable and scalable KPSs. An example of a non scalable KPS is the "basic" KPS where for a network size of N, the KDC chooses $P = \binom{N}{2}$ secrets and each node is assigned $k = N - 1$ secrets. The basic KPS is *not* susceptible to collusions. In other words, secrets exposed from devices $A_1 \cdots A_n$ will not reveal any information about secrets between any two nodes B and C where $B, C \notin \{A_1, A_2, \ldots A_n\}$. However, with a limitation of $\mathbb{O}(n)$ storage, only a network size of n is feasible.

While the ever increasing capabilities of storage implies that even network sizes of several tens of millions can be very easily realized using a nonscalable KPS (after all 10 million 64-bit secrets requires a mere 80 MB of storage), one of the reasons why non scalable KPSs are still undesirable is that we ideally require a very large ID-space (160-bit IDs) even if the actual number of deployed nodes may be only millions (say 2^{24}). Without a priori knowledge of the IDs that will be deployed, predistribution of pairwise secrets for just the 2^{24} nodes is cumbersome. In such situations, whenever a new node joins a network, every existing node should be provided a secret corresponding to the new node.

A. Scalable KPSs

Scalable KPSs on the other hand (with the same limitation of $k = \mathbb{O}(n)$) can support *unrestricted* network sizes, but can only tolerate collusions of n nodes pooling their secrets together. Thus the scalability of scalable KPSs comes at the price of sacrificing their resistance to collusion. That tradeoffs between scalability and collusion resistance are possible was first realized by Blom et al [5] who proposed the first KPS in the literature.

A.1 Blom's Scheme

In the Blom's polynomial based KPS [5] the KDC chooses a $(k-1)$-degree symmetric polynomial in two variables over $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$,

$$f(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{ij} x^i y^j \bmod q, \quad x, y, a_{ij} \in \mathbb{Z}_q. \tag{3}$$

where $a_{ij} = a_{ji}$ are $P = \binom{k}{2}$ independent secrets chosen by the KDC from $\mathbb{Z}_q$ Every node is assigned a *unique* ID from $\mathbb{Z}_q$. A node A (node with public ID[1] $A \in \mathbb{Z}_{\shortparallel}$) re-

[1]Thus q should be a 128-bit or 160-bit prime in order to facilitate ID-based allocation of secrets.

ceives $g_A(x) = f(x, A)$ securely. Note that $g_A(x)$ has k coefficients. These k correspond to k secrets assigned to node A by the KDC. Now two nodes A and B can calculate $K_{AB} = K_{BA} = f(A, B) = f(B, A) = g_A(B) = g_B(A)$ independently.

By compromising $\lceil\frac{k-1}{2}\rceil$ nodes an attacker can construct P equations required to solve for the P unknowns (the secrets chosen by the KDC) and compromise the KPS completely. However as long as $n \leq \lceil\frac{k-1}{2}\rceil$ nodes are compromised, the attackers cannot reveal *any* "extra" secret (ie. secrets other than those revealed from the compromised nodes). The failure of Blom's KPS thus occurs catastrophically. Blom's scheme requires about $k \approx 2n$ keys in each node to be n-secure.

A.2 Probabilistic Assurances

More generally, KPSs can be considered as (n, p)-secure, where an attacker who has exposed all secrets from n nodes can expose only a fraction p of all pairwise SAs. Deterministic KPSs (like the two schemes by Blom) can be seen as special cases of probabilistic KPSs (PKPS) where p takes only binary values - 0 or 1. For PKPSs p is a continuous function of n. More specifically, for probabilistic KPSs p is a monotonically increasing function of n where $0 \leq p(n) \leq 1$.

For a network size of N there are $\binom{N}{2}$ possible pairwise SAs. With n' compromised nodes, ideally we would wish that $\binom{N-n'}{2}$ SAs should still be safe. For deterministic KPSs all $\binom{N-n'}{2}$ SAs are safe (with probability 1) as long as $n' \leq n$. However, for $n' > n$, *all* $\binom{N}{2}$ SAs (and all P secrets of the KDC) can be compromised by the attacker. In the case of PKPSs, an attacker who has pooled together all secrets from n nodes can expose a fraction p of the $\binom{N-n}{2}$ SAs. Unlike deterministic KPSs (where the failure of the KPS occurs catastrophically) PKPSs fail gracefully (with increasing n).

A.3 LM-KPS

Leighton and Micali [6] were the first to propose a KPS (LM-KPS) with probabilistic assurances. In the LM-KPS, defined by two parameters (k, L), the KDC chooses a set of $P = k$ secrets $\{K_1 \cdots K_k\}$, and each node is provided with a set of k secrets. The k secrets provided to each node are repeatedly hashed versions (uniformly distributed between 1 and L times) of the secrets chosen by the KDC. Thus node A is provided with secrets

$$\mathbb{S}_A = \{K_1^{\bar{a}_1}, K_2^{\bar{a}_2}, \ldots, K_k^{\bar{a}_k}\}, 1 \leq \bar{a}_i \leq L \qquad (4)$$

where $K_i^j = h^j(K_i)$ represents the result of successive hashing of K_i, j times using a secure hash function $h()$. The parameter L is the "maximum hash depth." For LM-KPS, the *upper* bound for k is $\mathbb{O}(n^3)$ [6], [15].

A.4 Random Allocation of Subset (RAS)

Many KPS based on random allocation of a subset (RAS) of k keys to every node from a pool of P keys chosen by the KDC, have been proposed. In such schemes the SA (shared secret) between any two nodes is a function of all KPS secrets that are *common* to both nodes. Such schemes are extensions of techniques in [7] - [9] that relied on *deterministic* strategies for allocation of subsets of keys to every node. Dyer et al [10] were was the first to point out the simplicity and effectiveness of *random* subset allocations. Dyer's strategy has also been used for broadcast authentication [11], and more recently, in the context of securing sensor networks [12].

In ID-based RAS schemes [13], [14] the indices of the secrets assigned to every node is tied to the node ID through a one-way function. For example in [14] a public function

$$F(A) = \{A_1, A_2, \ldots A_k\}, 1 \leq A_i \leq P, 1 \leq i \leq k \qquad (5)$$

determines the k of the P indices of secrets assigned to node A. Two nodes A and B can determine the intersecting indices (corresponding to the secrets both A and B possess) by computing $F(A) \cap F(B)$, and use all common secrets to derive the SA K_{AB}.

If we desire a scheme for which $p(n') < p(n) = \delta$, $\forall n' < n$, and also desire to minimize the number of keys k assigned to every node, the optimal choice of $\xi = k/P = \frac{1}{n+1}$, for which $k = \mathrm{e}(\mathrm{n}+1)\log(1/\mathrm{p})$ [15]. As any two nodes share only $\xi k = m = \mathrm{e}\log(1/\mathrm{p})$ secrets (on an average), only m of the k secrets are *used* for evaluation of any SA. Note that irrespective of the size of n, the number of secrets m that will be used for evaluating any SA is the same (as m does not depend on n). However, to determine the m intersecting indices of A an B, both nodes have to evaluate $F(A) \cap F(B)$ - the complexity of which is $\mathbb{O}(k)$.

Extensions of RAS schemes with LM-KPS [6] - or hashed random preloaded subsets (HARPS) have also been proposed [15]. For HARPS [15] $F(A)$ generates k hash depths (between 1 and L) in addition to the k indices.

III. A New Class of KPSs

In the "basic" KDS, for a network of N nodes the KDC chooses $\binom{N}{2}$ secrets and each node is provided with $N - 1$ secrets. The primary problem with the basic KDS is that it does not scale very well. A trivial scalable extension of basic KDS is to employ several such independent "small scale" deployments of basic KDSs in parallel.

A. MBK

In the "multiple" basic KDS (MBK), m such independent deployments, each catering only for a network size of M, are used. Together, the m systems cater for practically unrestricted network sizes. We will assume that each node is assigned a b-bit ID (say $b = 128$ or 160), which we shall refer to as the "long ID." Corresponding to the long ID of each node, are m "short-IDs," each $\log_2(M)$ bits long. More specifically, for a node A (node with long ID A) a simple public one-way function

$$f(A,i) = a_i, 1 \leq i \leq m, \tag{6}$$

where a_is are uniformly distributed[2] between 1 and M is employed to determine the short-IDs (of node A). Similarly the m short IDs of node B will be determined by $b_i = f(B,i), 1 \leq i \leq m$.

The KDC chooses m sets of secrets $\mathbb{S}^1 \cdots \mathbb{S}^m$, where *each* set consists of $\binom{M+1}{2}$ secrets[3]

$$\mathbb{S}^i = \{K_i(j_1,j_2)\}, 1 \leq i \leq m, 1 \leq j_1, j_2 \leq M, \tag{7}$$

and $K_i(j_1,j_2) = K_i(j_2,j_1)$. Now a node A with short IDs $a_1, a_2, \ldots a_m$, is provided with M secrets from *each* of the m systems - corresponding to the short-ID in each system. Specifically, node A with short IDs is provided with $k = m \times M$ secrets

$$\mathbb{S}_A = \{K_i(a_i,j)\}, 1 \leq i \leq m, 1 \leq j \leq M. \tag{8}$$

Node B, likewise, is provided with $k = Mm$ secrets $\mathbb{S}_B = \{K_i(b_i,j)\}$.

Nodes A and B can now independently discover m shared secrets $S_i = K_i(a_i,b_i) = K_i(b_i,a_i)$. Note that by executing $f(A,i)$ and $f(B,i)$ (m times, for $1 \leq i \leq m$) both nodes can determine their respective short IDs $\{(a_i,b_i)\}$ in all m schemes. All S_i "elementary shared secrets" are used to derive the pairwise secret K_{AB}.

A.1 Analysis

MBK provides probabilistic assurances. Firstly note that there is a finite probability that two nodes with different long IDs may be assigned the same set of m short IDs ($\log_2 M$-bits each). The probability of this event however is very low - the same as the probability of collision in a $m \log_2 M$-bit hash function. For example, for $m = 64, M = 1024$ this probability is a miniscule $2^{-64 \times 10/2} = 2^{-320}$. What is of greater concern is the ability of an attacker who has managed to extract all mM secrets from n nodes. Let us define by $p(n)$ the probability that an attacker who has pooled together all secrets from n nodes (that does not include A and B), can determine K_{AB}.

Note that the secret $K_i(a_i,b_i)$ is not unique to nodes A and B. There is a probability that some node C (in the attacker's pool of compromised nodes) may also have the secret $K_i(a_i,b_i)$, *if* $f(C,i) = c_i \in \{a_i,b_i\}$. As the output of the one-way function $f()$ is uniformly distributed between 1 and M, the probability that a node C has access to $K_i(a_i,b_i)$ is

$$\Pr\{c_i \in \{a_i,b_i\}\} = \gamma = \frac{2M-1}{M^2} \approx 2/M, \tag{9}$$

where the approximation holds for large M. The probability that an attacker who has access to all secrets from n nodes (that does not include A and B) *cannot* discover $S_i = K_i(a_i,b_i)$ is $\epsilon(n) = (1-\gamma)^n$. Thus the probability $p(n)$ that the attacker *can* compromise *all* m such S_is shared by A and B (and thus determine K_{AB}) is

$$\begin{aligned} p(n) &= (1-\epsilon(n))^m = (1-(1-\gamma)^n)^m \\ &= \approx (1-\mathrm{e}^{-n\gamma})^m \approx (1-\mathrm{e}^{-\frac{2n}{M}})^m \end{aligned} \tag{10}$$

where the first approximation follows from the well known identity $(1-1/x)^x \approx \mathrm{e}^{-1}$ for large x.

B. Choice of Parameters m and M

Due to the freedom in choice of the parameters m and M there are many possible choices (of combinations of m and M) to achieve the requirement. To understand the implication of different choices, we should first consider the effect of the parameters on the nature of the complexity introduced (computation, storage etc.).

1. The storage required by each node is $k = mM$.
2. The computational complexity of the public function is $\mathbb{O}(m)$. More specifically, each node has to perform $2m$ computations of the public function $f()$.
3. The number of symmetric cipher operations for evaluating a pairwise SA is also m.

In any scenario where multiple secrets are used, it is common practice to encrypt all secrets using a single master secret, as protecting a single master secret may be easier. Such encrypted secrets can now be stored in unprotected storage locations (say pluggable storage or even storage accessible over a network). Whenever the secrets have to be used however, they have to be fetched from storage, decrypted, and used. As an example many home devices could store their secrets (encrypted) in a desktop computer and use unprotected wireless channels to access the required secrets (whenever necessary), from the desktop computer.

As yet another example, in wireless sensor networks sensors could be deployed with all secrets stored in a base station. The secrets of a sensor A (stored in the base station) is encrypted using a secret that is known only to node A. Compromise of the base station will not reveal the secrets of node A (even if the base station stores the secrets of all nodes a single-point of failure is avoided). In such a scenario, while a sensor may be *assigned* a large number ($k = mM$) of secrets, a sensor with ν neighbors may require to *use* at most νm secrets. The νm secrets alone could be fetched from the base station.

[2] The function $f()$ could be a pseudo-random bit stream generator, where each instance of execution of $f()$ calls for generation $\log_2 M$ bits. Alternately it could be the last $\log_2(M)$ LSBs of a hash function operating on the values A and i.

[3] Apart from the $\binom{M}{2}$ secrets of the form $K_i(j_1,j_2), j_1 \neq j_2$, the system also includes M secrets of the form $K_i(j_l,j_l), 1 \leq l \leq M$. Note that $\binom{M+1}{2} = \binom{M}{2} + M$.

In situations where the secrets have to be fetched over a wireless network the value m has obvious implications on wireless bandwidth. Even in situations where the secrets are stored in pluggable storage it is always beneficial to reduce m. The number of secrets (m) to be fetched from the storage media, decrypted and used (for evaluation of any SA), has significant implications on the complexity. The total number of stored secrets $k = mM$ however may have very little impact. Thus while we would like to achieve the desired (n, p)-security (or $p(n) < \delta$ where δ is sufficiently small) with as little storage $k = mM$ as possible, it is indeed advantageous to reduce m to as low a value as possible, *even if it implies increasing* storage $k = mM$.

Seeing that $m = k/M$, we can reqwrite Eq (10) as

$$k = \frac{2n\log(1/p)}{-x\log(1-\mathrm{e}^{-\mathrm{x}})}, \text{ where } x = \frac{2n}{M}. \quad (11)$$

Minimizing k calls for maximixing $-x\log(1-\mathrm{e}^{-\mathrm{x}})$, which occurs when $\mathrm{e}^{-\mathrm{x}} = 1/2$, or $x = \log(2)$. Now substituting $\mathrm{e}^{-\frac{2\mathrm{n}}{\mathrm{M}}} = 1/2$ in Eq (10), we have

$$\begin{array}{lclcl} p(n) & = & \left(\frac{1}{1-1/2}\right)^m & = & \frac{1}{2^m} \\ m & = & \log_2(1/p) & = & \log(1/p)/\log(2) \\ M & = & 2n/\log(2) & \approx & 2.885n \\ k & = & mM = \frac{2n\log(1/p)}{(\log(2))^2} & \approx & 4.16n\log(1/p) \end{array} \quad (12)$$

If we desire to reduce m by a factor a (to $m' = \frac{m}{a}$) we need $\mathrm{e}^{-\frac{2\mathrm{n}}{\mathrm{M}}} = 1 - 1/2^{\mathrm{a}}$, to achieve the desired $p(n)$. Thus we need to increase M' (and therefore $k' = M'm'$) to

$$\begin{array}{rcl} M' & = & 2n/\log(1-1/2^a) \\ k' = m'M' & = & mM\dfrac{\log(1-1/2)}{a\log(1-1/2^a)} \end{array} \quad (13)$$

The table below depicts the tradeoffs involved in reducing m (by a factor a) vs the corresponding increase in the storage $k = mM$ (or the factor $\frac{k'}{k} = \frac{\log(1-1/2)}{a\log(1-1/2^a)}$ in Eq (13)):

a	2	3	4	5	6	8
k'/k	1.204	1.730	2.685	4.366	7.336	22.137

In other words, decreasing m by a factor of 3 (from the value that minimizes k for the desired $p(n)$) would require increasing storage k by a factor 1.73 (or increasing M by a factor 3×1.73). In many scenarios this may be an advantageous trade-off as the value of m more directly translates to use of resources than k, especially if storage is inexpensive.

B.1 Numerical Example

As a numerical example, an MBK scheme with parameters $M = 2048$, $m = 64$ is $(n = 710, p = 2^{-64})$-secure (or $p(n) = 2^{-64}$). In other words, an attacker who has exposed $k = mM = 131,072$ secrets each from $n = 710$ nodes, can discover one in 2^{64} pairwise SAs. Note that even discovering *which* SAs *can* be compromised is computationally intractable for an attacker (for such low values of p). The resources demanded by such an MBK system are

1. a mere 1 MB of storage (assuming that each of the k secrets are 64-bits long);
2. $m = 64$ of the $k = 131,072$ secrets to be fetched from storage, and used in symmetric cipher operations to discover the pairwise SA;
3. $2m$ executions of a pseudo random function $f(.,.)$ which returns a $\log_2 M$-bit (11-bit) uniformly distributed random number.

If we desire to increase n by a factor 64 (or $n > 45000$, and $p(n) < 2^{-64}$), all we need to do is to increase M by the same factor to $M = 131072 = 2^{17}$ (keeping $m = 64$), for a total storage requirement of 64 MB. Note that the increase in computational complexity is insignificant - now $f(.,.)$ generates $\log_2(M) = 17$-bit random numbers instead of 11-bit random numbers, and the complexity of symmetric cipher operations is still the same as earlier.

If we desire to reduce m to 32 we need to increase M to 315,621 (resulting in a storage requirement of 81 MB). In practice it is convenient to make M a power of 2 (for efficient implementation of the public function $f()$). So we could choose $M = 2^{18}$ and $m = 36$ to meet the same requirement ($p(45000) < 2^{-64}$), calling for a storage of slightly over 75 MB. Alternately we could choose $M = 2^{19}$ and $m = 24$, calling for 96 MB storage.

Recall that the security of PKPSs reduce gracefully when n increases. For the scheme with $M = 2^{19}, m = 24$ for instance, an attacker has to compromise all secrets from over 132,000 nodes to discover one in 2^{32} (4 billion) SAs. To discover one in 2^{20} (a million) SAs the attacker has to expose all secrets from over 216,000 nodes.

C. Deployment of MBK

In a practical realization of MBK upto m independent KDCs could be employed. A node A may be provided with one unique secret for every KDC. Each KDC may deliver M KPS secrets to the node (encrypted with the unique secret) either over a wired network or even optical storage media. The node A could then extract each secret, and store it in some convenient media (like flash card) which could be plugged into any mobile / handheld device. Apart from a secret stored in the mobile device used for encrypting the secrets stored in the flash memory card, the secrets could also be encrypted with a password of the user. Thus an attacker may not be able to access secrets from stolen devices (flash cards). In other words, for compromising 50,000 nodes, an attacker needs to enlist 50,000 willing colluders. Even in very large networks (perhaps hundreds of millions of nodes) assimilating such a large number of willing colluders may be infeasible for any attacker.

Each KDC can utilize a different master secret M_i and a strong one-way hash function $h()$. All KDCs however agree on the public function $f(.,.)$. To assign M secrets to node A, the KDC i computes $a_i = f(A, i)$ and proceeds to compute M secrets

$$K_i(a_i, j) = h(M_i, a_i, j), 1 \leq j \leq M \tag{14}$$

that will be delivered to A. Note that even if M is several millions, any desktop class computer performing the job of the KDC should require mere seconds to compute all keys for a node. Apart from reducing the resources of each KDC, using m independent KDCs can also reduce concerns regarding abuse of powers by a single all powerful escrow.

D. MBK vs Other KPSs

D.1 Blom's KPS

To see why Blom's scheme is not very practical for resisting large collusions, consider the overheads required for a 50000-secure Blom's scheme. In such a scheme the KDC may choose a master secret M and generate each of $\binom{100000}{2} \approx 5$ *billion* secrets on demand, or just store the 5 billion secrets as the storage required for the KDC is not prohibitive. Each node is provided with $k = 100,000$ secrets. In order to compute the secrets (coefficients of the polynomial) for node A the KDC has to perform over 5 billion finite field operations. This is obviously expensive.

While the storage required for 100,000 secrets for each node is not very high[4] (less than 2 MB) the computational complexity is very high. For calculating a pairwise shared secret each node has to fetch *all* its 100,000 secrets and perform 100,000 finite field operations. Even using asymmetric cryptographic primitives may involve less computational complexity than Blom's scheme if we desire large n.

D.2 RAS

For RAS schemes [15]

$$p(n) = (1 - \xi(1 - \xi)^n)^k \tag{15}$$

where $\xi = k/P$. The optimal choice of ξ that minimizes k, should maximize $\xi(1-\xi)^n$, which is $\xi^* = 1/(n+1) \approx 1/n$ (for large n). Once again making use of the identities $(1 + 1/x)^x \approx \mathrm{e}$ for large x and $\log(1-y) \approx -y$ for $y << 1$, we can easily see that $k \approx n\mathrm{e}\log(1/\mathrm{p})$ for $\xi = 1/n$. The average number of shared secrets between any two nodes is $m = \mathrm{e}\log(1/\mathrm{p})$. MBK on the other hand requires $k = Mm \approx 4.16n\log(1/p)$ for the choice of $M = 2.885n$. In other words, MBK requires storage larger by a factor $\frac{4.16}{\mathrm{e}} \approx 1.53$ compared to RAS. Thus RAS scheme to meet the requirement $p(45000) \approx 2^{-64}$ would require only $k \approx 5482750$ (a little less than 42 MB storage). RAS would require $m = 121$ keys to be fetched and $m = 121$ symmetric cipher operations.

[4]All ID and secrets are values from Z_q, thus calling for 20 bytes for each value as q is a 160-bit prime.

TABLE I
Comparison of Blom's polynomial (BP) scheme, RAS and MBK.

KPS	Total Storage (k)	m	CPF	CKDC
BP	$2n$	$2n$	None	$\mathbb{O}(k^2)$
RAS	$(n+1)\mathrm{e}\log(1/\mathrm{p})$	$\mathrm{e}\log(1/\mathrm{p})$	$\mathbb{O}(k)$	$\mathbb{O}(k)$
MBK	$4.16n\log(1/p)$	$\mathrm{e}\log(1/\mathrm{p})$	$\mathbb{O}(m)$	$\mathbb{O}(k)$

Furthermore, unlike MBK where it is possible to reduce m (by increasing k) for RAS schemes m cannot be reduced below $\log(1/p)$ to achieve (n,p)-security. Reducing m below the optimal value (for a desired (n,p)-security, that minimizes k) implies reducing ξ and increasing k such that $m = \xi k$ is reduced. It is easy to see that m cannot be reduced below $\log(1/p)$ even if $k \to \infty$. For large k and $\xi << 1/n$, $\xi(1-\xi)^n \approx \xi(1 - n\xi) \approx \xi$ for $\xi << 1/n$. Thus

$$\begin{aligned} p(n) &= (1 - \xi(1-\xi)^n)^k = (1-\xi)^{m/\xi} \\ &\approx \left((1-\xi)^{1/\xi}\right)^m \approx \mathrm{e}^{-\mathrm{m}}. \end{aligned}$$

In other words, even when $\xi \to 0$ and $k \to \infty$, $m = \xi k$ *cannot* be reduced below $\log(1/p)$ to ensure (n,p)-security.

However, the biggest drawback of RAS schemes comes from the complexity of the public function $F(A) \cap F(B)$ that have to be computed by nodes A and B to determine the $m = 121$ shared indices. Computing this would involve pseudo-random generation of $2k$ (*over 10 million*) uniformly distributed random numbers. While this may still be far less expensive than Blom's scheme, the complexity of the public function renders RAS schemes unsuitable for large n.

Table I provides a ready comparison of different KPSs in terms of the relationships between 1) total storage (k), 2) number of secrets that need to be used for evaluating and SA (m) and 3) complexity of public functions (CPF) and 4) Complexity for KDC (CKDC) for a desired level of collusion resistance, (n,p).

Blom's scheme is perhaps unsuited for scenarios demanding n even of the order of hundreds, as for larger n the computational complexity may be comparable to that of asymmetric schemes. RAS schemes on the other hand may be unsuitable for n greater than a few thousands as the complexity of the public function becomes expensive. The only bottle-neck for MBK (with $\mathbb{O}(\Bbbk)$ complexity) is storage.

IV. Hashed MBK

A simple extension of the MBK scheme, the hashed MBK (HMBK) can improve the performance of MBK for the same M and m. Specifically, HMBK realizes improvements over MBK by realizing an increase in n for the same $p(n)$. HMBK is actually a combination of MBK and the KPS with probabilistic assurances proposed by Leighton and Micali, LM-KPS [6] defined by two parameters (k, L) (see Section II-A.3, page).

HMBK is defined by three parameters (M, m, L). As in MBK, the KDC chooses m sets of secrets[5] $\mathbb{S}^1 \cdots \mathbb{S}^m$, where each set consists of $\binom{M+1}{2}$ secrets. All such secrets could also be generated from a single master secret M_i chosen by the KDC.

However, the public function $f()$ now produces two values - a $\log_2 M$-bit value and a $\log_2 L$-bit value, where L is the maximum hash depth of the keys. For example, for $M = 1024$ and $L = 64$, $f(A, i) = [a_i \parallel \bar{a}_i]$ returns a 16 bit integer - the first 10 bits determines the short-ID $1 \leq a_i \leq M$, and the last six bits determine the hash depth, $1 \leq \bar{a}_i \leq L$. The secrets assigned to node A are now

$$\begin{aligned} \mathbb{S}_A &= \{K_i^{\bar{a}_i}(a_i, j)\}, \text{ where} \\ K_i^{\bar{a}_i}(a_i, j) &= h^{\bar{a}_i}(K_i(a_i, j)) \end{aligned} \tag{16}$$

for $1 \leq i \leq m$, $1 \leq j \leq M$. In other words a secret $(K_i(a_i, j)$ is repeatedly hashed $\bar{a}_i$ times before it is assigned to A as $h^{\bar{a}_i}(K_i(a_i, j))$.

As in MBK, any two nodes can determine m elementary shared secrets. Recall that in MBK, for nodes A and B (with $f(A, i) = a_i$ and $f(B, i) = b_i$), the secret corresponding to index i is $K_i(a_i, b_i) = K_i(b_i, a_i)$. For HMBK however, node A has secret $K_i^{\bar{a}_i}(a_i, b_i)$ and node B has secret $K_i^{\bar{b}_i}(a_i, b_i)$ (where $f(B, i) = b_i \parallel \bar{b}_i$). The elementary shares S_i, $1 \leq i \leq m$ are now computed as

$$S_i = K_i^{\max(\bar{a}_i, \bar{b}_i)}(a_i, b_i). \tag{17}$$

If $\bar{a}_i > \bar{b}_i$, node B has to repeatedly hash its secret $K_i^{\bar{b}_i}(a_i, b_i)$, repeatedly, $(\bar{a}_i - \bar{b}_i)$ times. This yields a common elementary shared secret $S_i = K_i^{\bar{a}_i}(a_i, b_i)$ between A and B (for index i, where $1 \leq i \leq m$). Likewise, if $\bar{b}_i > \bar{a}_i$, node A will have to hash its secret $K_i^{\bar{a}_i}(a_i, b_i)$, $(\bar{b}_i - \bar{a}_i)$ times. For each of the m indices, the shared secret between A and B is at a hash depth $\max(\bar{a}_i, \bar{b}_i)$.

A. Analysis

In MBK a node C with $f(C, i) = c_i \in \{a_i, b_i\}$ has access to the secret S_i corresponding to index i between A and B. However, for HMBK an additional condition has to be satisfied - which is $\bar{c}_i \leq \max(\bar{a}_i, \bar{b}_i)$. For any i if we define

$$\begin{aligned} p_l &= \Pr\{\max(\bar{a}_i, \bar{b}_i) = l\} = \frac{2l - 1}{L^2} \\ q_l &= \Pr\{c_i \leq l\} = l/L, 1 \leq l \leq L \end{aligned} \tag{18}$$

the probability ϵ' that any $S_i = K_i^{\max(\bar{a}_i, \bar{b}_i)}(a_i, b_i)$ is safe from an attacker (who has exposed all mM secrets from n nodes) is

$$\epsilon' = \sum_{j=1}^{L} p_l(1 - q_l\gamma)^n = \sum_{j=1}^{L} \frac{2l - 1}{L^2}\left(1 - \frac{l\gamma}{L}\right)^n \tag{19}$$

[5]Or m KDCs choose one set of secrets each.

The probability $p(n)$ for HMBK is then

$$p(n) = (1 - \epsilon'(n))^m \tag{20}$$

As a *first order approximation*, it is easy to see that the expected value of l is $\bar{l} = \frac{2L}{3}$. Thus

$$\epsilon'(n) \approx \left(1 - \frac{2\gamma}{3}\right)^n \approx (1 - \gamma)^{3n/2} = \epsilon(3n/2), \tag{21}$$

implying that HMBK can tolerate 3/2 times as many compromised nodes as MBK for the same value of m and $k = mM$.

The computational overheads for HMBK (compared to MBK) is that in each of the m operations with secrets, a few repeated hash operations ($L/6$ on an average[6]) have to be performed.

B. Performance Comparisons

Figure 1 depicts plots of n vs $-\log_2(p)$ for MBK and HMBK. For MBK the parameters of m and M are chosen to achieve $p(n) < 2^{-64} \forall n < 45000$. Obviously, for $n > 45000$ $p(n) > 2^{-64}$. The y-axis value is $-\log_2(p) = 64$. A y-axis value of 32 indicates that an attacker can expose one in 2^{32} (or 4 billion) pairwise SAs. The plots are provided for two choices of m - $m_1 = 64$ and $m_2 = 24$, with the corresponding choice of M as $M_1 = 2^{17}$ and $M_2 = 2^{19}$ respectively. It is instructive to note that the design with lower m actually performs better for n larger than the design value ($n = 45000$) while larger m "over-performs" by a larger margin for $n < 45000$. This is not surprising as the choice lower m (and consequently higher M) is designed to be "storage optimal" for $n = 4 \times 45000$.

The figure includes 2 plots for HMBK with the same choices of m and M for $L = 64$. Note that HMBK (for the same m and M) meets the design criteria ($p < 2^{-64}$) even for $n = 70000$ (an improvement of a factor 3/2 as seen in the first order approximation.) Apart from being more efficient than MBK, HMBK also boasts a more graceful degradation of security with increasing n. This property is unfortunately not evident in the first-order approximation ($\epsilon'(n) \approx \epsilon(3n/2)$). The plots are calculated using Eq (20) (the first order approximation is *not* used for computing the plots).

V. Conclusions

More often the efficiency of any KPS is regarded as the ratio the achievable collusion resistance to the number of keys that need to be stored by each node. In practical application scenarios, we argued that the number of keys that need to be stored has a much lower impact on "resources"

[6]Note that for each of the m shared secrets only one node has to hash forward, on an average $L/3$ times as the expected value of the difference between two randomly chosen hash depths between 1 and L is $L/3$.

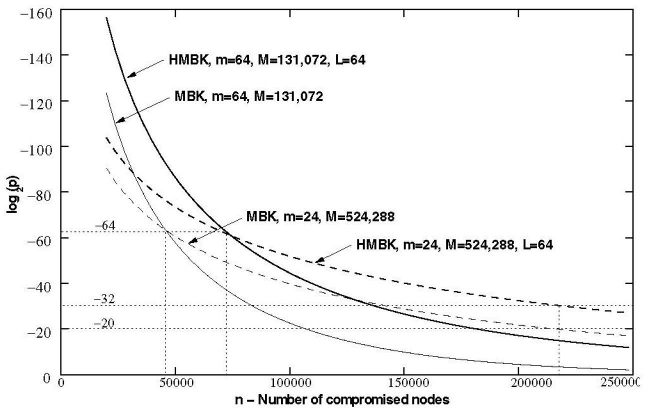

Fig. 1. Performance of MBK and HMBK for $m_1 = 64, M_1 = 2^{17}$ and $m_2 = 24, M_2 = 2^{19}$. The thick lines are plots for HMBK and the thin lines, MBK. The dashed lines are for $m = 24$ and the continuous lines for $m = 64$. The values for m and M have been chosen to guarantee $p(45000) < 2^{-64}$ for MBK. For HMBK $L = 64$. For HMBK with $m = 24$, $p(220000) < 2^{-32}$.

than the number of secrets to be used for evaluation of any security association, and the computational complexity. For most existing KPSs the bottle-neck for increasing the collusion resistance is offered by computational complexity. For the proposed schemes the computational complexity is trivial. The bottleneck is only available storage, which facilitates realization of KPSs with high collusion resistance as storage is the least expensive of all resources.

References

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

[2] D. Boneh, M. Franklin, "Identity-based encryption from the Weil pairing," Advances in Cryptology – Crypto'2001, Lecture Notes on Computer Science 2139, Springer-Verlag (2001), pp. 213–229.

[3] R. Dutta, R. Barua, P. Sarkar, "Pairing-Based Cryptography : A Survey," Cryptology ePrint Archive, Report 2004/064.

[4] D. Clarke, J-E Elien, M. Fredette, A. Marcos, R.L.Rivest, "Certificate chain discovery in SPKI/SDSI," Journal of Computer Security, vol. 9, no. 4, pp. 285 – 322(38), 2001.

[5] 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.

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

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

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

[9] 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.

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

[11] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," INFOCOMM'99, 1999.

[12] 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.

[13] 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.

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

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