# Broadcast Encryption Using Probabilistic Key Distribution and Applications

Mahalingam Ramkumar
Department of Computer Science and Engineering, Mississippi State University
Email: ramkumar@cse.msstate.edu

***Abstract*— A family of novel broadcast encryption schemes based on probabilistic key pre-distribution are proposed, that enable multiple sources to broadcast secrets, *without* the use of asymmetric cryptographic primitives. A comprehensive analysis of their efficiencies and performance bounds are presented. The paper also suggests a framework for applications of broadcast encryption, depending on 1) the relationship between the size of the network (or the total number of deployed devices) and the group size, and 2) the nature of the devices (stateless or otherwise) in the deployment. The utility of the proposed broadcast encryption schemes is investigated for securing content distribution applications based on the publish-subscribe paradigm.**

***Index Terms*— broadcast encryption, stateless vs stateful, probabilistic key predistribution**

## I. Introduction

Broadcast encryption (BE) [2] provides a means of establishing shared secret between $g$ privileged nodes, among of a set of $G = g + r$ nodes, where the $r$ nodes which are not provided with the secret are usually referred to as *revoked* nodes.

BE schemes involve a set-up phase where secrets are distributed to all nodes in the network. To disseminate a broadcast secret $K_b$ to all nodes (except $r$ specifically excluded nodes), the source 1) encrypts $K_b$ using $n$ keys $K_{e1} \cdots K_{en}$, and 2) broadcasts $n$ values[1] $K_{ei}(K_b), 1 \leq i \leq n$. The secrets $K_{e1} \cdots K_{en}$ are chosen in such a way that none of the $r = G - g$ nodes can (using the secrets they possess) determine *any* of the keys $K_{e1} \cdots K_{en}$. However, the privileged $G - r$ nodes should possess, or can determine using the secrets they possess, *at least one* of the secrets $K_{e1} \cdots K_{en}$, and thereby gain access to the secret $K_b$.

### A. Group Secrets

The capability to establish and control access to group secrets has a wide variety of applications like digital rights management (DRM) [3], publish-subscribe systems [4] and multicast communications [5]. For instance, in DRM applications regulating access to content $C$ is realized by encrypting content with a *content encryption key* $K_C$. The content encryption key is encrypted with the group secret $K_G$ (and $K_G(K_C)$ distributed with the content) to ensure that only members of the group can gain access to $K_C$, and hence the content.

*1) Protecting Group Secrets:* The ability to protect any secret depends on the number of nodes that have access to the secret. Obviously, the higher the number of nodes with access to a secret, the higher is the susceptibility of the secret to exposure. With BE, *all* except a few revoked nodes share the group secret. The use of weak security associations calls for assurances of *trustworthiness* of the entities that are provided with access to the secrets. In other words, entities provided with group secrets are *trusted not to reveal* the group secrets to unauthorized entities.

In most practical application scenarios involving group secrets, some proactive measures are required to protect the group secrets even from the end-users. For instance, the group secrets (and the pre-distributed secrets used for disseminating the group secrets) could be protected by a trustworthy device like a smart-card or a trustworthy chip housed in a DRM enabled device (DED). Providing assurances of trustworthiness of components that protect and use the group secrets (on behalf of a user or a DED) calls for physical shielding of such components from intrusions aimed at modifying their behavior, or exposing secrets. An unfortunate side effect physical shielding is reduced ability to dissipate heat [6]. If we can reduce the complexity of the protected components to *very low levels*, we can eliminate the need for proactive approaches for heat dissipation, thereby facilitating inexpensive *and* fool-proof shielding techniques. In other words, the complexity, and hence computational ability, of components that have to be protected, can have a significant effect on the *cost* of such components. Thus limiting such components to employ only symmetric cryptographic primitives can be a useful strategy.

### B. Multi-source Broadcast Encryption

Many efficient BE schemes that utilize only symmetric cryptographic primitives have been proposed in the literature since the first BE scheme by Fiat and Noar [2]. Most solutions [7] - [9] are tree-based, where the source of the broadcast is assumed to be the key distribution center (KDC) who distributes the secrets in the first place.

This paper is based on “On Broadcast Encryption with Random Key Pre-distribution Schemes,” by M. Ramkumar, which appeared in the Proceedings of the 1st Intl. Conf. Information Systems Security - ICISS 2005, Kolkata, India, Dec. 2005 [1].

This work was supported in part by NSF grants EIA-0103722 and CNS-0448070.

[1] In the rest of this paper we use the notation $K(M)$ to represent encryption of a quantity $M$ using a secret $K$, using some standard symmetric cipher.

However, BE schemes that cater for broadcasts by multiple sources have some very useful applications. Many multi-source BE schemes employing public key primitives have been proposed [10] - [11]. Furthermore, Naor et al [7] have indicated that at least some tree-based schemes can be readily extended to cater for multi-source BE *if* asymmetric cryptographic primitives are employed. On of the several advantages of BE using probabilistic key predistribution schemes (PKPS) discussed in this paper, is that they cater for BE by multiple sources *without* the use of asymmetric cryptographic primitives.

In Section II we provide a overview of tree-based (T-BE) and PKPS based BE schemes (PKPS-BE) and discuss extensions of the schemes to facilitate broadcasts by multiple sources. A comprehensive analysis of the efficiencies of PKPS-BE schemes and bounds on their performance are discussed in Section III.

In application scenarios involving a single source, the number of deployed devices, or the *network size* $N$, is the same as the *group size* $G$, controlled by the single source. On the other hand, in scenarios involving multiple sources, different sources may control and regulate access to their group secrets. Thus several (possibly overlapping groups) of varying group sizes $G << N$ can exist within the network. In Section IV we discuss some of the fundamental requirements of BE schemes to support the two models - 1) $N = G$ model, and 2) $N >> G$ models. In Section V we elucidate why PKPS-BE is ideally suited for $N >> G$ models, and discuss a publish-subscribe system involving multiple broadcast sources. Conclusions are offered in Section VI.

## II. Broadcast Encryption Schemes

Many tree-based BE schemes (T-BE) have been proposed in the literature [7] - [9]. However, in this paper we shall restrict ourselves to the complete-subtree scheme proposed by Noar et al [7].

### A. Tree-based Schemes

In the complete-subtree scheme for a system with $N = 2^L$ devices, the $N$ devices are assumed to correspond to the leaf nodes of a binary tree of depth $L$. Associated with each of the $2N-1 = \sum_{i=0}^{L} 2^i$ nodes in the binary tree, $n_{ij}, 0 \leq i \leq L, 0 \leq j \leq 2^i - 1$ are $2N-1$ secrets $K_{ij}$, chosen by the KDC.

Apart from the one-to-one correspondence of the $N$ devices $I_j, 0 \leq j \leq N-1$ in the system with the leaf nodes $n_{Lj}, 0 \leq j \leq N-1$, each device is also associated with $L$ direct ancestors - one in each level of the tree. A device $I_l$ receives $L+1$ secrets - $L$ secrets associated with its $L$ ancestor nodes, and the secret $K_{Ll}$ corresponding to the leaf node $n_{Ll}$.

Figure 1 exhibits such a tree for $L = 3$ (or $N = 8$). Device $I_3$ corresponds to the node $n_{33}$ with ancestors $n_{21}$, $n_{10}$ and $n_{00}$, receives secrets $K_{33}$, $K_{21}$, $K_{10}$ and $K_{00}$. All devices receive $K_{00}$, half the devices receive $K_{10}$, and so on.

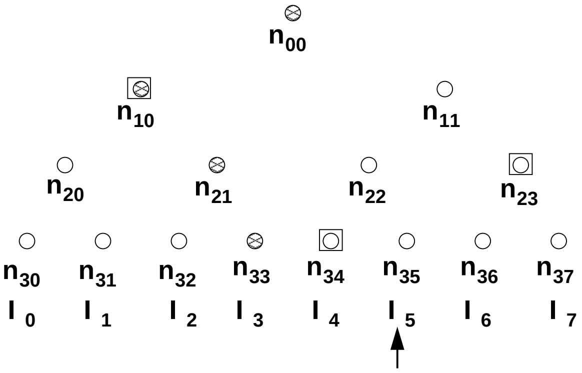

Figure 1. The binary tree used in the complete-subtree scheme. The nodes filled with patterns associated with device $I_3$. Keys corresponding to nodes enclosed in boxes are used for revoking $I_5$.

To revoke $I_5$ the KDC encrypts a broadcast secret $K_b$ with 3 secrets - $K_{34}$, $K_{23}$, and $K_{10}$, and broadcasts

$$\mathcal{B} = [I_5 \parallel (K_{34}(K_b), K_{23}(K_b), K_{10}(K_b))]. \quad (1)$$

Any device receiving the broadcast $\mathcal{B}$ can determine which secrets have been used by the source (as the revoked node is explicitly specified). Devices $I_0 \cdots I_3$ can decrypt $K_{10}(K_b)$. $I_6$ and $I_7$ can decrypt $K_{23}(K_b)$. $I_4$ can decrypt $K_{34}(K_b)$. In general, revoking any device will call for $L$ encryptions of the broadcast secret. Note that any number of devices can be revoked together, in one broadcast. Further, while revoking 1 device calls for using $\log_2 N$ encryptions, revoking more than one, say $r > 1$ devices, will require *less* than $r \log_2 N$ encryptions. Noar et al [7] have shown that even in the *worst case scenario* only $r \log_2(N/r)$ encryptions are called for.

*1) Multi-Source Extensions:* Extending T-BE schemes to cater for broadcasts by multiple sources calls for interpreting the value assigned to each of the $2N-1$ nodes in the binary tree as a *public* value, corresponding to which secrets (or private keys) are assigned to every device. Thus corresponding to each of the $2N-1$ nodes $n_{ij}$, the KDC generates public-private key pairs $\{(U_{ij}, R_{ij})\}$.

Each device stores $L+1$ private keys. In this case device $I_3$ stores private keys $R_{33}$, $R_{21}$, $R_{10}$ and $R_{00}$. The public values $U_{ij}$ of all $2N-1$ nodes are made public (provided to all potential sources of broadcasts). Thus any source with knowledge of the public keys can encrypt the broadcast secret using the public keys, which only devices with the corresponding private keys can decrypt.

### B. Probabilistic Key Distribution

Most probabilistic key predistribution schemes (PKPS) are based on the strategy of allocating a subset of $k$ keys to each device, from a pool of $P$ keys chosen by the KDC. Gong and Wheeler [12], Mitchell and Piper [13] and Dyer et al [14] have investigated various strategies for allocation of subsets, motivated by Erdos et al's [15] seminal work on uniqueness of subset intersections. Dyer et al [14] (1995) also pointed out that complex deterministic allocation strategies can be easily replaced with simple *random* allocation strategies with very little penalty. Dyer's strategy of random allocation has also found applications in broadcast authentication [5]. In this paper, random subset allocation strategies as random preloaded subsets (RPS) as in [16]. Due to random allocation of secrets, RPS provides probabilistic assurances.

The first key predistribution scheme with probabilistic assurances was however proposed by Leighton and Micali (LM) [17] in 1993. Unlike RPS where each device is asssigned a subset of the KDC's secrets, in LM-KPS each device receives a repeatedly hashed version of *all* $P = k$ KDC's secrets. In hashed random preloaded subsets (HARPS) [18], Ramkumar et al proposed a generalization of RPS and LM-KPS, where the subset of allocated secrets are successively hashed versions of the KDC's secrets.

*1) RPS and HARPS:* In $(P, k)$ RPS, the KDC chooses a set of $P$ keys $\mathbb{S} = \{K_1 \cdots K_P\}$, and a public random function $F()$. A device with ID $A$ is assigned $k$ indexes determined by

$$F(A) = \{A_1, A_2, \ldots A_k\}, 1 \leq A_i \leq P, \; A_i \neq A_j \forall i \neq j.$$

Corresponding to the public indexes (as $F()$ is public anyone can evaluate $F(A)$ to determine the indexes of secrets with device $A$), device $A$ is provided with secrets

$$\mathbb{S}_A = \{K_{A_1}, K_{A_2}, \ldots, K_{A_k}\}. \qquad (2)$$

In $(P, k, L)$ HARPS, the KDC chooses $P$ keys $\mathbb{S} = \{K_1 \cdots K_P\}$, a cryptographic hash function $h()$, and two public function $F()$ and $f()$. For a device $A$, the indexes $\{A_1 \cdots A_k\}$ are assigned as in RPS. However, each index assigned to a device is also associated with a "hash depth" $a_i = f(A, A_i)$, $1 \leq i \leq k$, $1 \leq a_i \leq L \forall i$, uniformly distributed between 1 and $L$. The $k$ secrets assigned to device $A$ are now

$$\mathbb{S}_A = \{{}^{a_i}K_{A_i} = h^{a_i}(K_{A_i})\}, 1 \leq i \leq k, \qquad (3)$$

where $h^j(K_l) = h(h(\cdots j\text{times}(K_l)\cdots))$ represents the result of applying $j$ successive hashes (using the cryptographic hash function $h()$) on the value $K_l$. Note that any entity with key ${}^dK$ can determine ${}^xK$ for $x \geq d$ (but not for $x < d$). As earlier, both $F()$ and $f()$ are public. So anyone can determine the indexes (and the hash depth) of secrets assigned to any device.

*2) BE Using HARPS:* As HARPS is a generalization of RPS (RPS is equivalent to HARPS with $L = 1$) we shall begin by illustrating the principle behind broadcast encryption using HARPS. First, we shall restrict ourselves to BE by the KDC, and then discuss simple extensions to cater for efficient broadcast of secrets by multiple sources.

Consider the illustrative example depicted below with $P = 8, k = 4, L = 4$. The KDC chooses $P = 8$ keys $K_1 \cdots K_8$. Device $A$ has keys with indexes $i = 1, 2, 4, 6$ at hash depths 4, 2, 1 and 3 respectively - or keys ${}^4K_1, {}^2K_2, {}^1K_4$ and ${}^3K_6$. The row marked $d_i$ is the hash depths the KDC can *safely* employ for each $1 \leq i \leq P$ for encrypting $K_b$. For example, for revoking $A$ and $B$ the KDC can use keys ${}^3K_1$, ${}^1K_2$, ${}^2K_5$, ${}^1K_6$, ${}^4K_7$, and ${}^4K_7$ as none of the secrets can be determined by $A$ or $B$, or even by $A$ and $B$ pooling their secrets together. Note that while $A$ has a secret ${}^4K_1$ corresponding to index 1, $A$ cannot determine the preimage (under the hash function $h()$) ${}^3K_1$. Device $C$ can determine ${}^4K_7$ by hashing its secret for the index, ${}^4K_7$, twice, and thus decrypt $K_b$. Device $D$ can determine ${}^3K_1$ or ${}^1K_6$ or ${}^4K_8$.

In the case of RPS, hashing is not employed - thus each device either has a key corresponding to some index, or does not. In this case the KDC can choose the key corresponding to indexes 7 and 8 to encrypt the broadcast secret (to revoke $A$ and $B$). As it turns out in this case, both $C$ and $D$ can employ the key with index 7.

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $A$ | 4 | 2 | x | 1 | x | 3 | x | x |
| $B$ | x | 3 | 1 | x | 3 | 2 | x | x |
| $d_i$ | 3 | 1 | x | x | 2 | 1 | 4 | 4 |
| $C$ | x | 3 | 4 | 1 | x | x | 2 | x |
| $D$ | 2 | x | x | x | 3 | 1 | 4 | x |

*3) BE by Sources Other Than KDC:* One of the main advantages of PKPS-BE schemes comes from the fact that they trivially cater for BE by any source[2]. The KDC can authorize *any* source, say a content distributor $\Theta$, to perform broadcast encryption by providing the distributor $\Theta$ with $PL$ "encryption secrets,"

$$\mathfrak{S}_\Theta = \{{}^jK_j^\Theta = h({}^jK_i \parallel \Theta)\}, 1 \leq i \leq P, 1 \leq j \leq L, \quad (4)$$

or in the case of RPS, just $P$ secrets $\mathfrak{S}_\Theta = \{K_i^\Theta = h(K_i \parallel \Theta)\}$. It is important to note that the encryption secrets $\mathfrak{S}_\Theta$ (which are used to encrypt the broadcast secret) reveal no information about the KDCs secrets or even the "decryption" secrets (the $k$ secrets assigned to any device), as long as the hash function is pre-image resistant.

To revoke $r$ device, just as the KDC can use a subset (or hashed subset) of the secrets $K_1 \cdots K_P$ - say ${}^{d_1}K_{I_i} \cdots {}^{d_n}K_{I_n}$, to encrypt the broadcast secret $K_b$, the external source $\Theta$ can use ${}^{d_1}K_{I_1}^\Theta \cdots {}^{d_n}K_{I_n}^\Theta$ to encrypt $K_b$. A device that has a decryption secret ${}^dK_i$ can still compute the corresponding encryption secret ${}^{d'}K_i^\Theta = h({}^{j'}K_i \parallel \Theta)$ for any $d' \geq d$. Thus the efficiency of BE, whether performed by the KDC or some other source (with $PL$ encryption secrets), is the same.

Canetti et al [5] suggested a similar technique - involving encryption secrets assigned to external sources by the KDC (which reveal no information about the KDCs secrets or the decryption secrets) to facilitate broadcast *authentication* by external sources.

## III. PERFORMANCE OF PKPS BE

The efficiency of PKPS-BE is the same irrespective of whether the source is the KDC with $P$ secrets, or some external source provided with $LP$ secrets derived from

[2] Recall that while T-BE schemes can be extended to cater for broadcast by any source, such extensions call for the use of expensive asymmetric primitives.

the $P$ secrets of the KDC. In the rest of this section, we shall therefore assume that the source is the KDC.

### A. *Efficiency*

The KDC has access to all secrets $K_1 \cdots K_P$ (at hash depth 0, or $K_i = {}^0K_i$). Each of the $r$ (to-be-revoked) devices have $k = \xi P < P$ keys each (or $\xi = k/P < 1$ ). The hash depths of their keys are uniformly distributed between 1 and $L$.

Now consider a key indexed $i$, which say $u$ of the $r$ revoked devices possess. Let the hash depths of those $u$ keys be $d_1 \cdots d_u$, with $d' = \min(d_1 \cdots d_u)$. In other words, the union of keys of the revoked devices include ${}^{d_1}K_i, {}^{d_2}K_i, \ldots, {}^{d_u}K_i$ (considering only secrets for index $i$). The KDC can still safely employ key ${}^{d'-1}K_i$ to encrypt the broadcast secret.

Let $n_j$ be the average number of such safe keys that the KDC can use at hash depth $1 \leq j \leq L$. With these $n = \sum_{j=1}^{L} n_j$ encryptions of the broadcast secret, the KDC hopes to "reach" (or convey to secret $K_b$ to) *every* privileged device.

It is easy to see that for $j = L$, the $n_j$ keys correspond to the keys that none of the $r$ devices have (at *any* hash depth). The probability that any device has a key indexed $i$ is $\xi = \frac{k}{P}$. Thus the probability that none of the $r$ devices have key $i$ is $(1 - \xi)^r$. In other words, $n_L = P(1 - \xi)^r$.

For RPS (with $L = 1$) $n = n_L = P(1-\xi)^r$ is the *total* number of safe keys. For HARPS, let us now evaluate the expression for $n_j$ for a general $j$. As any key is assigned to any node with probability $\xi = k/P$, it can be easily seen that the probability that *exactly* $u$ of $r$ nodes have a secret corresponding to some index $i$ is the binomial probability $B_\xi(r, u) = \binom{r}{u}\xi^u(1 - \xi)^{(r-u)}$.

Let us represent by $\nu_{ij}$, the probability that the KDC employs hash depth $j$ for key indexed $i$. Recalling that for any index $i$, the KDC will employ depth $j$ if $d' = \min(d_1 \cdots d_u) = j + 1$, and recognizing that

$$\begin{aligned}\Pr\{d' = j+1\} &= \Pr\{d' > j\} - \Pr\{d' > j+1\} \\ &= \frac{(L-j)^u - (L-j-1)^u}{L^u},\end{aligned}$$

it can be readily seen that

$$\begin{aligned} n_j &= P\nu_{ij}, \text{ where} \\ \nu_{ij} &= \sum_{u=1}^{r} B_\xi(r,u)\Pr\{d' = j+1\} \end{aligned} \qquad (5)$$

*1) Outage Probability:* While it is guaranteed that none of the $r$ devices (even if they pool all their secrets together) can decipher the broadcast secret, there is a possibility that some of the $g = G - r$ privileged devices may not be able to decrypt *any* of the $n = \sum_{i=1}^{L} n_j$ encryptions. Let $p_o$ be the "outage probability" - the probability that an arbitrary device among the group of $g$ privileged devices, cannot decrypt *any* of the $n$ encryptions.

In order to decrypt a secret encrypted with key index $i$ at depth $j$, the node should have the secret indexed $i$ at depth $d \leq j$, which will occur with a probability $\frac{\xi j}{L}$. Obviously, encryption keys corresponding to higher hash depths are more useful in conveying the secret to more privileged nodes. Thus for a particular encryption key at hash depth $j$ (or any one of the $n_j$ keys) the probability of outage is $p_{o_j} = (1 - \xi\frac{j}{L})$.

In general, the KDC may not have to use all the $n = \sum_{j=1}^{L} n_j$ possible safe keys. Only a subset $n_e = \sum_{j=q}^{L} n_j$ keys may be used to achieve a target outage probability of $p_o^*$. For instance the KDC will first try to use only keys at depth $L$ (as they will be more useful for more privileged nodes), and if necessary consider using keys at depth $L - 1$ and so on. In general, the source may use all possible safe keys with depth greater than $q$, and $n_q{}' \leq n_q$ keys at depth $q$. In this case, the probability of outage for any device, and hence the total number of encryptions $n_e^*$ required to convey the broadcast secret to all privileged nodes are

$$p_o^* = (1 - \xi\frac{q}{L})^{n'_q}\prod_{j=q+1}^{L}(1 - \xi\frac{j}{L})^{n_j}, \qquad (6)$$

$$n_e^* = gp_o^* + n'_q + \sum_{j=q+1}^{L} n_j, \qquad (7)$$

where the term $gp_o^*$ accounts for the accidentally missed devices. For instance, if the KDC chooses a target of $p_o^* \approx 1/g \approx 1/G$, one of the $g$ devices will be accidentally missed *on an average*, for every revocation. To reach the missed devices, either an additional safe key can be added for each missed device (as the source does not typically use all possible safe keys to achieve the target of $p_o$), or they can be conveyed by encrypting the broadcast secret with a unique key provided to each device.

*2) Overheads:* Apart from the broadcasting several encryptions of the secret, recall that for T-BE schemes the broadcast should indicate the IDs of the revoked nodes (for example $I_5$ in Eq (1)). For PKPS-BE, while this is possible, it is more efficient, both in terms of bandwidth needed and computational complexity at the receiver, to instead provide the *indexes* and the hash depths of the keys used to encrypt the broadcast secret. If we represent $\mathbb{H}(x) = -x\log_2(x)$, the expression for the overheads $o_I$ and $o_D$ (in number of bits) for conveying the indexes and hash-depths respectively (of secrets used) are [1]

$$o_I = P\left\{\mathbb{H}\left(\frac{n_e}{P}\right) + \mathbb{H}\left(\frac{P - n_e}{P}\right)\right\} \qquad (8)$$

$$o_d = n_e\sum_{j=q}^{L}\mathbb{H}\left(\frac{n_j}{n_e}\right). \qquad (9)$$

### B. *Performance Bounds*

The exact analytical expressions for the relationship between $P$, $k$, $L$ and the number of encryptions $n_e^*$ (Eqs (5) - (7)) necessary to revoke $r$ devices, can be used readily for evaluating the performance of PKPS-BE for various choices of $P, k, L$ and $r$. However, they provide very little intuition regarding the bounds of performance.

To gain some more insight we shall look more closely at the simpler case (RPS) where no hashing employed.

With $P$ secrets chosen by the KDC, and $k = \xi P$ provided to each device, the minimum achievable probability of outage for revoking $r$ devices, is

$$p_{min} = (1-\xi)^n = (1-\xi)^{P(1-\xi)^r} \approx 1/G_{max}, \qquad (10)$$

where $G_{max}$ is the maximum possible group size. Obviously $p_{min}$ can be reduced to any extent (or $G_{max}$ increased to any extent) by increasing $P$. The question now is what is the "optimal" choice of $\xi$ for some $G, r$?

*1) Minimizing P:* The optimality however depends on *what* we are trying to minimize. For instance, if we wish to minimize $P$, the optimal choice of $\xi$ should minimize $\mathbf{C} = (1-\xi)^{(1-\xi)^r}$, which occurs when $\xi \approx 1/r$ (for large $r$). Also, for large $r$

$$(1-1/r)^r \rightarrow \mathrm{e}^{-1} \Rightarrow \mathrm{p_{min}} = (1-\xi)^{\mathrm{P/e}}. \qquad (11)$$

Making use of the fact that $log(1-\xi) \approx -\xi$ for small $\xi$ (or large $r \approx 1/\xi$), we thus have

$$P \approx \mathrm{e}r\log\lambda \quad k \approx \mathrm{e}\log\lambda \quad n_e \approx r\log\lambda. \qquad (12)$$

where $\lambda = p_{min}^{-1} \approx G_{max}$. For a network size of one billion ($2^{30}$), where we would desire $p_{min} \approx 2^{-30}$, for $r = 128$ we require $k \approx 57$, $P = rk = 7296$, calling for $n_e \approx 2661$ encryptions of the broadcast secret.

*2) Minimizing $n_e$:* More often, optimality of BE schemes is measured in terms of the number of encryptions, $n_e$, required for revoking $r$ devices. Now instead of choosing $\xi = 1/r$, let us instead choose $\xi = a/r, a > 1$. In this case $p_{min} = (1-a/r)^{P(1-a/r)^r} \approx (1-a/r)^{\mathrm{Pe^{-a}}}$, or

$$P \approx r\frac{\mathrm{e^a}}{a}\log\lambda \quad k \approx \mathrm{e^a}\log\lambda \quad n_e \approx \frac{r}{a}\log\lambda. \qquad (13)$$

In other words, if we increase $P$ by a factor $\mathrm{e^{a-1}/a}$, and $k$ by a factor $e^{a-1}$, we can reduce the bandwidth needed for conveying the broadcast secret by a factor $a$. For $a = 4$ for instance, $P = 36636, k = 1145$, but $n_e$ reduces to 665 encryptions for revoking 128 devices (for a group size of 1 billion).

### C. Over-provisioning Keys

We saw that for a group size of $G = 2^{30}$ (or $p_o = 1/G$) RPS with parameters $P = r\mathrm{e}\log\mathrm{G} = 7296, \mathrm{k} = \mathrm{e}\log(\mathrm{G}) = 57$ can revoke $r = 128 = P/k$ device, using $r\log G$ encryptions of a broadcast secret. With this choice of parameters

① of the $P = r\mathrm{e}\log\mathrm{G}$ possible secrets of the KDC, only a fraction $P/\mathrm{e}$ are "safe" (on an average), when $r$ devices have to be revoked;

② all $n_e = r\log G$ safe secrets are *required* to achieve the target outage probability of $p_o = G^{-1}$. Thus even if $r$ is less than 128, the KDC will still need to transmit $n_e = 1/\xi\log G$ encryptions in order to convey the secrets to the $G - r \approx G$ privileged nodes (or achieve outage probability $p_o \approx G^{-1}$).

In other words, for $r < 128$ the bandwidth efficiency per revoked node reduces (or $n_e/r$ increases, as $n_e$ remains the same), and for $r$ much larger than 128, the system is unusable.

However, now consider a scenario where the same $P = 7296, k = 57$ scheme is used for a group size of $G = 2^{10}$ (thousand, instead of a billion). In this case we are actually employing $P$ and $k$ 3 times larger than what is required to revoke 128 devices[3]. Alternately, we can interpret this approach as a scheme corresponding to the choice of $a = 2.11$ (as $\mathrm{e^a}\log(2^{10}) = 57$) to reduce bandwidth by a factor $a$ (see Eq (13)), and designed for $r^* = a * 128 \approx 270$ and $G = 2^{10}$.

Thus for a group size of $G = 2^{10}$ the $(P = 7296, k = 57)$ scheme can revoke upto 270 devices with $n_e/r = (\log G)/a = 3.29$ encryptions per revoked device. At the same time, 128 devices can be revoked with an efficiency of $n_e/r = \log G = 6.93$, as only a fraction of the $P/\mathrm{e}$ safe secrets need to be employed for ensuring outage probability $p_o \approx \log(G^{-1})$.

Thus while a system designed to minimize $P, k$ for some $r$ is efficient only for a narrow range of $r$, *by over-provisioning keys we can realize efficient operation over a wider range* of $r$. As an other example, RPS with $P = 200 \times 128, k = 200$ can cater for efficient operation for a range of $r$ between 128 and 340 for $G = 2^{30}$, and a range of 128 to 460 for $G = 2^{20}$ (a million). For larger ranges, we could increase $k$ further, or alternately, employ *parallel deployments* of RPS with different values of $\xi = k/P$, so that together, they can be used efficiently for a wide range of $r$.

### D. HARPS vs RPS

Without over-provisioning keys, the KDC cannot revoke much more than $P/k$ devices in a batch as the KDC *runs out* of the $n = P(1-\xi)^r$ safe secrets that can be used to convey the broadcast secret to the privileged devices in the group. However, in the case of HARPS, in addition to $n_L = P(1-\xi)^r$ safe secrets (corresponding to which none of the $r$ nodes have a secret assigned) other $n_{L-1}, \ldots n_1$ safe secrets are available. For example, if two revoked devices have the secret corresponding to index $i$, say ${}^{42}K_i$ and ${}^{26}K_i$ (where $L = 64$), the KDC can use the secret ${}^{25}K_i$ (potentially useful to $\xi\frac{(G-r)25}{64}$ privileged devices).

While closed form expressions for the performance bounds of HARPS (akin to Eqs (12) and (13) for RPS) are not readily tractable, the performance of HARPS can still be evaluated using the analytical expressions derived in the previous section (see Eqs (5) - (7)), for the relationships between $n_e$, $r$, $(P, k, L)$ and $G \approx 1/p_o$.

Figure 2 depicts the performance of RPS and HARPS with the same $P = 7296, k = 57$ (and $L = 64$ for HARPS) in terms of $n_e/r$ ($y$-axis) and the number of revoked devices, $r$ ($x$-axis), for a group size of $G = 2^{30}$. Note that till the point $r$ is not much larger than 128, both RPS and HARPS perform identically, as only keys at hash depth $L$ are used for HARPS (keys at lower hash depths are not needed *yet*). But for larger $r$ while RPS

[3] Choosing $k = 19, P = 2432$ would suffice.

runs out of safe secrets, HARPS can begin using keys at lower hash depths, and thus continue to operate efficiently (even for $r > 500$ as can be seen from the figure)

Just as over-provisioning helped improve the usable range of $r$ for RPS, it can also help to *further* improve the usable range of HARPS. For example, for $P = 25600, k = 200$, for values of $L = 1, 2, 4, 6, 8, 16, 32$ and 64 respectively, the range of usable[4] $r$ is between 128 and $r_L$, where $r_L$ for different $L$ is shown in the table below (the case $L = 1$ corresponds to RPS):

| $L$ | 1 | 2 | 6 | 8 | 16 | 64 |
|---|---|---|---|---|---|---|
| $r_L$ | 340 | 565 | 1040 | 1185 | 1520 | 2000 |

Now consider a scenario where we desire to cater for efficient revocation for a range of $r$ from 128 to say 1000, for a group size of 1 billion. With RPS, we can realize this by using two schemes, one with $(k = 200, \xi = 1/128)$, and the second with $(k = 200, \xi = 1/360)$. However (as can be seen from the table above) a single deployment of HARPS with $k = 200, \xi = 1/128, L = 6$ can meet this requirement (usable range of $r$).

In Figure 3 the plot labelled RPS depicts the performance (in terms of $n_e/r$ for different $r$) when the two RPS schemes are used in parallel (the first is used for $r \leq 155$, and second for $r > 155$). The plot labelled HARPS, $L = 6$ is for the case with a single HARPS deployment ($P = 25600, k = 200, L = 6$), caters for a slightly larger range of $r$ than two deployments of RPS used in parallel. As a quick comparison of the two approaches, 1) HARPS requires 200 secrets to be assigned to each device, RPS requires 400; 2) the KDC requires $P = 25600$ secrets for HARPS, and $25600 + 360 * 200 = 97600$ for RPS; 3) to facilitate BE by external sources, each external source requires $25600 \times L = 153600$ ($L = 6$) encryption secrets for HARPS, 97600 (the same as the KDC) for RPS.

*1) Choice of $P, k, L$ for Practical Deployments:* A reasonable approach then, to cater for efficient revocation for a wide range of $r$, may be to use say four independent deployments, 1) $\xi = 1/4, k = 100$, 2) $\xi = 1/16, k = 100$, to cater for small $r$, 3) $\xi = 128$ for $r$ between 128 and 1050, and 4) $\xi = 1024, k = 200$ for larger $r$ - for a total of 600 keys per device.

Further, the scheme supporting large $r$ ($\xi = 1/1024$) could use large $L$ (say 512) to facilitate batch sizes even upto 30,000. However, depending on the storage ability of the external source, the source does *not* have to store all $L \times 1024 \times 200$ encryption secrets for the scheme with $\xi = 1/1024$. For instance, if the external source decides to acquire only $1024 \times 200$ secrets at hash depth $L$, the deployment of HARPS can still be used with the same efficiency as RPS by the external source (while at the same time the KDC can employ much larger batch sizes). Note that in this case the external source has, and can thus use only the $n_L = P(1 - \xi)^r$ safe secrets. Or the situation is no different from using RPS instead.

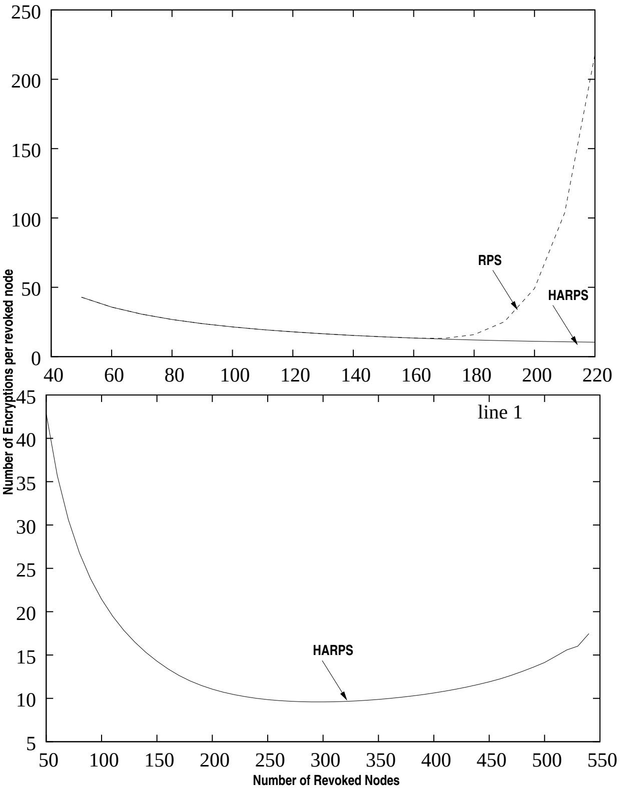

Figure 2. Comparison of HARPS and RPS for $P = 7296, k = 57$. For HARPS $L = 64$.

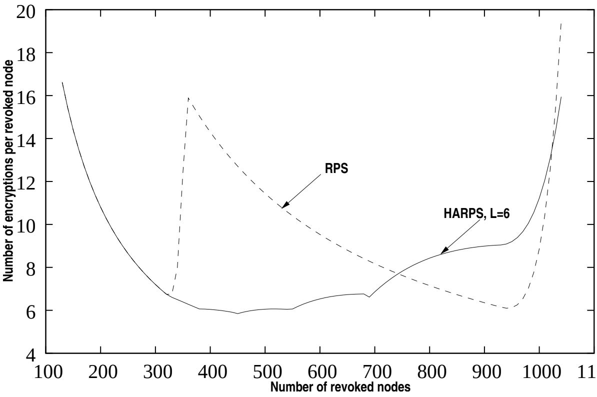

Figure 3. Comparison of two approaches - one using two parallel deployments of RPS and the second using one deployment of HARPS with $L = 6$.

Thus HARPS can simultaneously be used in the "RPS mode" by external sources that cannot afford to store a large number of secrets. Such a scheme (HARPS used in "RPS mode") can still be used by external sources for efficient revocation of up to 3000 devices (while the KDC can revoke upto 30,000 devices using $\sum_{j=1}^{L} n_L$ safe secrets). Furthermore, if the source can store $2 \times 1024 \times 200$ encryption secrets (say corresponding to hash depths $L$ and $L/2$), the external source can support batch sizes upto 5000 nodes (in this case, for the external sources the scheme is equivalent to HARPS with $L = 2$).

By providing 200 more keys to each node of a $P = 25000 \times 200, k = 200, L = 1024$ (or $1/\xi = 25000$) HARPS scheme, the KDC can support batch sizes of up

[4] We define the usable range $r_0 = 1\xi$ to $r_L$, where for revoking $r_L$ devices the efficiency is the same as revoking $r_0 = 1/\xi$ devices. The maximum efficiency (or minimum $n_e/r$) occurs at some $r$ between $r_0 = 1/\xi$ and $r_L$.

to a *million*. While this calls for a storage of 5 million keys by the KDC, in practice the KDC does not have to actually *store* all secrets - it could simply generate any secret *on demand* using a single (or a few) highly protected secrets using strong hash functions.

Thus with HARPS, there is *almost no practical limitation* on the maximum batch size for the KDC. External sources, can also support sufficiently large batch sizes $r$ with storage of $\mathbb{O}(r \log G)$. In most practical scenarios, even several GBs of storage is an inconsequential requirement for external sources (for example distributors of digital content who may have to deal with thousands of terabytes of content). Furthermore, as we shall argue in the next section, while it is desirable for the KDC to support large batch sizes, it is *not* really necessary for revocation by external sources to support large batch sizes (while it is still desirable for revocations by KDC).

## IV. Models for BE

From the perspective of the KDC, the "network size" $N$ is the number of devices that are assigned (or *could* be assigned) secrets. Most conventional models for BE assume that the group size $G$ is the same as the network size $N$. Furthermore, the devices taking part in such deployments are also assumed to be *stateless* devices [7]. In other words, once keys are distributed to such devices, there is no way to provide them with new secrets. In most cases, the source of the broadcast (the entity which revokes devices) is also the entity that distributes the secrets in the first place - the KDC. On the other hand, there are many application scenarios calling for BE, where $N >> G$. In such scenarios many independent sources (apart from the KDC) will be able to control group secrets for *their* specific interest groups consisting of perhaps $G << N$ users / devices.

### A. $G = N$ Models

A practical example of a $G = N$ stateless model is the case of DVD content protection, where each DVD player is provided with a set of secrets (that cannot be modified during the lifetime of the device). By default, all DVD players can render all DVDs, unless explicitly revoked. The content in a DVD is encrypted with a content encryption key $K_C$, and the secret $K_C$ is encrypted with a secret $K_N$. The secret $K_N$ is disseminated using BE (included in every DVD) so that only non-revoked devices can gain access to $K_N$, and hence $K_C$, and thus decrypt the content. Typically, DVD players that are suspected to have been compromised by attackers are revoked. More specifically, a player is "compromised" when an attacker has exposed (or is suspected to have exposed) secrets from the player.

By using secrets exposed from one or more compromised players, say $D_1 \cdots D_n$, an attacker can *synthesize* any number of illegitimate players. If such illegitimate players are discovered, it may be possible to employ traitor tracing [7] schemes to determine the original DVD players $D_1 \cdots D_n$, whose secrets were employed for constructing the illegitimate players. Thus revoking $D_1 \cdots D_n$ will simultaneously revoke *all* such illegitimate DVD players in addition to the $D_1 \cdots D_n$.

*1) Revocation in $G = N$ Stateless Models:* Assume that a month after such a system is deployed, $n_1$ devices have been identified as compromised. For all content distributed from this point onwards, a new group secret $K_{N_1}$ is chosen and conveyed to all devices except the $n_1$ devices. At the end of the second month, say $n_2$ more compromised devices are identified. Now a new group secret $K_{N_2}$ is chosen and conveyed to all but $n_1 + n_2$ devices (in all DVDs pressed after this point; revoked devices can still play older DVDs).

*2) $G = N$ Stateful Models:* In $G = N$ *stateful* models, the DVD players can remember (store) changing group secrets. In this case, it may appear at first sight that revocations can be performed in batches. For example, all devices share a secret $K_{N_0}$ initially. At the end of the first month, a broadcast revokes $n_1$ devices by providing a secret $K'_{N_1}$ to all other devices. Thus the new group secret shared (and stored) by all $N - n_1$ legitimate devices is $K_{N_1} = K_{N_0} \oplus K'_{N_1}$. At the end of the second month a revocation broadcast revokes $n_2$ devices by broadcasting $K'_{N_2}$ that the $n_2$ devices cannot decrypt. The group secret after the second revocation is then $K_{N_2} = K_{N_1} \oplus K'_{N_2}$.

Note that while the $n_1$ devices revoked in the first batch can still gain access to $K'_{N_2}$, they cannot gain access to the new group secret $K_{N_2} = K_{N_1} \oplus K'_{N_2} = K_{N_0} \oplus K'_{N_1} \oplus K'_{N_2}$ as they do not have access to $K'_{N_1}$. Unfortunately, a pirate with access to secrets from one device in the first batch and one device in the second batch can still gain access to the new group secret $K_{N_2}$. Thus *if* the purpose of revocation is for excluding devices suspected of key compromises, revocation should not be performed in batches.

### B. $N >> G$ Models

However, in many application scenarios where BE can be performed by multiple sources, the network size $N$ can be substantially larger than the group size $G$. Consider a scenario where a maximum of $N = 2^{32}$ (about 4 billion) DRM enabled set-top boxes (STB) could be deployed for playing protected video content. Every end-user owns one such STB. A content distributor $D$ may have $G << N$ paying *subscribers* (say $G = 2^{20}$, or a million). From the perspective of the distributor, the group size is a million. Each member of the group (or the STB's belonging to the $G$ users) may be provided with a secret $K_{G_0}$ as part of the subscription process. Later, when the distributor desires to cancel the membership of $r$ (say 1000) of his $G$ subscribers, the distributor can broadcast a new secret $K_{G_1}$ to the $G - r$ continuing subscribers, that explicitly revokes $r$ subscribers.

*1) Revocation by Sources Other than the KDC:* Thus the new secret that is broadcast, is protected only from the $r$ explicitly revoked subscribers. However, Both the $G - r$ continuing subscribers and the $N - G$ *non-members* can gain access to the secret $K_{G_1}$. To prevent any of the $N-G$ non-members from gaining access to the broadcast secret the entire broadcast may be encrypted with the group secret $K_{G_0}$. Nevertheless, it still does not prevent a revoked user from colluding with one of the $N-G$ users outside the group $G$ to determine the secret $K_{G_1}$.

In other words, ideally $N-(G-r) \approx N$ users / STBs will have to be revoked, which is obviously impractical (it is far more efficient to unicast the broadcast secret independently to each of the $G-r$ nodes when $N >> G$). However, mandating that all $N-(G-r)$ nodes be revoked by $D$, while impractical, is also *unnecessary*.

In the $N = G$ scenario, *and* for revocation broadcasts by the KDC for $N >> G$ models, revocation will occur when secrets are (or suspected of being) compromised. However, the purpose of revocation by external sources in $N >> G$ models is to control access to group secrets to paying customers. *A revoked user is not necessarily more malicious than a user who is not revoked, or some user outside the group*. Given the fact that it is impractical to revoke all $N-(G-r)$ devices in any case, mandating that revocation broadcasts by distributors like $D$ should not be batched, does not help much.

*2) Revocation by KDC:* However revocation by KDC in $N >> G$ models will still be for the same purpose as $N = G$ models - ejecting devices that are suspected of compromise of secrets. While ideally, revocation broadcasts by the KDC should be able to support unlimited $r$, in practice this is not an essential requirement if devices taking part in the deployment are *not stateless*.

Note that stateless devices are not well suited for $N >> G$ scenarios in any case as external sources will need to provide dynamic group secrets to members of the group. Consider a scenario where broadcasts by KDC supports batch sizes upto $r_{max} = 100,000$. Arguably, irrespective of the network size $N$, a scenario where such a large number of devices are suspected of being compromised, is a crying need for *renewal* of secrets. The devices revoked by the KDC will not be allowed to take part in renewal.

In other words, for systems that are *not* stateless, $r_{max}$ is just the number of devices that *trigger renewal* of the system. While we would still like $r_{max}$ to be high (as the process of renewal may be expensive), it is sufficient if $r_{max}$ is "high enough." In other words, while the BE by KDC *should* support large $r$, it does *not* have to cater for *unbounded* $r$.

### C. Batch Sizes for External Sources

For revocation by sources other than the KDC, where revocation *can* be performed in batches, at first sight it might seem that we could simply employ a scheme optimized to revoke *one* device in each batch, in which case *any* number of devices can be revoked efficiently. However, there are two very important reasons as to why this is not a good approach: 1) schemes optimized for low $r$ will employ $\xi$ very close[5] to 1 (or almost every node has almost every KPS secret) - and are thus less secure; 2) the efficiency increases (or $n_e/r$ reduces) for schemes optimized for larger $r$ (small $\xi$).

*1) Resistance to Synthesis Attacks:* By compromising secrets from a certain number of devices, an attacker can determine a large fraction of the secrets[6] of the system.

One measure of the security of any KDS is their resilience to "synthesis attacks." More specifically, if an attacker needs to compromise secrets from $n_s$ devices to expose *all* secrets of a fraction $p$ of the devices (or $pN$ devices that are not part of the $n_s$ compromised devices), the resistance of the KDS to synthesis attacks is $p(n_s)$. For RPS with $\xi = 1/128$ and $k = 200$, by compromising all secrets from $n_s \approx 340$ devices, an attacker can synthesize one in a million devices (or $p(340) \approx 10^{-6}$. HARPS performs significantly better under this metric. For $\xi = 1/128$ and $k = 200, L = 64$, for HARPS $p(1650) \approx 10^{-6}$. On the other hand, if $\xi = 200/206$ (RPS optimized for batch size of 1), even by compromising secrets of one node (or $n_s = 1$), the attacker has access to all secrets of one in every two thousand devices (or $p(1) = 1/2000$).

*2) Bandwidth Efficiency:* In practical application scenarios it is only the efficiency for large batch sizes that really matters. If a PKPS-BE scheme that is optimized for a batch size of 100 is used for revoking 2 devices, the overhead may be the same as the case of revoking 100 devices. The fact that the overhead is say 20 KB instead of 200 bytes may not however be a serious limiting factor. However, we would certainly desire that the overheads for revoking say 100,000 devices is not prohibitively high.

Let us consider two scenarios 1) HARPS with $k = 200$ optimized for batch size of one, and 2) HARPS optimized for $r >> 1$, for revoking $r$ nodes. In the first case the $r$ independent broadcasts (even though they can actually be sent togother) revoke one device each. In other words, each privileged device will receive $r$ secrets, while the revoked devices will receive only $r-1$ secrets. The final group secret then is derived from all $r$ secrets, thus shielding the secret from the $r$ revoked devices. However, in this case the overall outage probability for the privileged devices increases, as outage can happen even if *one* of the $r$ secrets "evade" a privileged device.

Thus instead of aiming for an outage probability of $p_o = G^{-1}$, with batched revocation (with batch size of one) our target is to ensure outage probability less than $p'_o$, for each batch, where $(1-p'_o)^r \approx G^{-1}$, or $p'_o \approx (rG)^{-1}$) instead. In other words, effectively the scenario is equivalent to increasing the group size $G$ by a factor $r$. We already know that PKPS can take advantage of reduced group sizes $G$ to improve their efficiency (irrespective of $N$). Obviously, the effective increase in

[5] For example, for $G = 2^{30}$, and a batch size of 1, with a limit of 200 keys assigned to each device, the best choice of parameters for RPS is $P = 206, k = 200$, and $P = 201, k = 200, L = 64$ for HARPS.

[6] Even while each device is provided with a unique secret (which will be used under conditions of outage), note that such secrets are meant to be used rarely.

group size for batched approaches will make them even less efficient.

The table below compares the achievable $n_e/r$ for two schemes - one with $\xi = 200/201 \approx 1, L = 64$ for $r = 1$, and the other with $\xi \approx 1/128, L = 64$ for $r = 500$, for three different group sizes $G$ (a billion, million and thousand). For the batched scheme with $\xi \approx 1$, the table also indicates the reduction in efficiency due to the need to cater for reduced outage probability. In other words, row 2 in the table is $n_e$ for revoking one device *without* taking the need to reduce outage probability into account. Row 3 (labelled $\xi \approx 1^*$) however, takes this into account.

| | $G = 2^{30}$ | $G = 2^{20}$ | $G = 2^{10}$ |
|---|---|---|---|
| $\xi = 1/128$ | 5.39 | 2.71 | 0.62 |
| $\xi \approx 1$ | 5.8 | 3.81 | 2.02 |
| $\xi \approx 1^*$ | 7.68 | 5.58 | 3.62 |

## V. Application: A Publish-Subscribe System

A publish-subscribe system [4] consists of a large number of users, who assume the role of publishers (of content), subscribers, or both. In other words, publishers control a group of subscribers, who are provided with a group secret, and hence access to the content published by the publisher. The ability to establish group secrets and regulate access to such group secrets is a very useful feature for any pub-sub system [19].

In such a system, every user may employ a smart-card, which protects 1) the KDS secrets used for disseminating group secrets, and 2) the group secrets themselves, from the end users. Content published by publishers could be encrypted directly or indirectly with the group secret, that only members of the group (legitimate subscribers) can access. For example, the content encryption secret $K_C$ could be encrypted with the group secret $K_G$ and distributed with the content.

In all scenarios the KDS secrets and group secrets have to be protected from the subscribers (only their smart-cards will have access to the secrets). In some application scenarios even $K_C$ will have to be protected (the secret $K_C$ may be handed over only to a trusted DRM enabled device). However in some scenarios (for example the content may be broadcast of stock-quotes encrypted with dynamic keys $K_{C_t}$ encrypted using the group secret) the session secret $K_C$ used for encrypting the content could be handed over to an untrusted software running on a subscribers desktop / laptop / PDA.

The role of the KDC is to identify compromised smart-cards and revoke such smart-cards from the deployment, by providing all non revoked smart-cards with a time varying universal secret $U_i$. Apart from encrypting session secrets (or content encryption secrets) with group secrets, all messages will also be encrypted using the secret $U_i$ to ensure that revoked devices cannot take part in the deployment. When the number of revoked devices crosses a threshold, the KDC could renew the KDS and provide new secrets to every smart-card in the system.

The publishers on the other hand do not have to concern themselves with the possibility of compromised devices. Furthermore, the system should also support mutual authentication of a publisher $A$ and potential subscriber $B$ (by facilitating discovery of a secret $K_{AB}$ that no other entity can discover), to facilitate initial dissemination of the publishers ($A$) group secret to the newly inducted subscriber $B$. Thereafter, revoking privileges of node $B$ (once $B$ cancels his subscription) can be realized by employing BE. In addition the system also need to cater for authentication of broadcasts (of both content and revocation messages).

### A. Desirable Features

A very desirable feature in such large scale application scenarios is practically unlimited scalability. Even while the total number of users (say $N'$) in the system may never exceed a few billions, it is still desirable to assign large IDs each user (for example 160-bit IDs) to facilitate ID-based approaches.

*ID-based Approaches:* For example, a smart-card belonging to a user described by a string $\mathbf{S}$ = "Alice B. Cryptographer, AnyTown, USA," could be conveniently assigned a 160-bit ID $A = h(\mathbf{S})$, where $h()$ is a secure hash function. Further in situation where it is necessary to bind a DRM enabled device, say with manufacturers serial number $MX - 435768AF23$, and owned by the user $A$ to the user $A$, such a device can be provided and ID $D = h(A \parallel MX - 435768AF23)$. Thus with ID based approaches with 160-bit IDs, even say $N' = 2^{60}$ IDs could be issued (or $N'$ devices deployed) with *very low* possibility of collisions. However the need for large ID-space is not for supporting very large network sizes (after all it is inconceivable that network sizes of the order of $2^{60}$ will ever be needed). The reason for large ID-space is to ensure that users cannot choose arbitrary pre-images to misrepresent themselves (as only their IDs are authenticated).

*Dynamic Group Sizes:* In practice $N'$ may be of the order of billions. The group sizes however can have a wide range. Even very large distributors may have only millions of subscribers. Thus it is desirable that the KDS operates efficiently for smaller group sizes, while still supporting any conceivable group or network size.

*Privacy:* As revocation broadcasts are accessible to any one, even users outside the system, it is undesirable for such broadcasts to explicitly indicate the identities of revoked nodes.

### B. PKPS-BE vs T-BE for Pub-Sub Systems

*1) Dynamic Group Size:* For the use of T-BE schemes for scenarios where $N >> G$, one possibility (though not very desirable) is to let the KDC control memberships of every group within the network (if we desire to eliminate the use of asymmetric primitives). Even in this case, irrespective of the group size $G$, the efficiency of revocation will still depend on the network size $N$. For example, for a T-BE scheme that caters for a network size of $2^{30}$ (a billion), where say $G = 2^{10}$ (a thousand) of the

$N = 2^{30}$ devices belong to a group, to revoke $r$ of the $G$ devices, the number of encryptions required is still $30 \times r$ - or $\log_2(N) = 30$ encryptions per revoked device. Unlike T-BE schemes, PKPS-BE can take advantage of reduced group sizes to increase their efficiency. Recall that for small group sizes (say 1000), the number of encryptions required per revoked device can be substantially smaller than 1.

*2) Scalability:* Even if we ignore the primary disadvantage of T-BE schemes for multi-source BE - the need for asymmetric cryptographic primitives, T-BE does not scale as well as PKPSs, due to the storage complexity to be borne by external sources. For multi-source T-BE schemes, sources other than the KDC need to store $\mathbb{O}(N)$ public values (more specifically $2N - 1$ public values corresponding to the $2N - 1$ nodes in the binary tree). This may not be a problem in practice even for network sizes of billions. After all, storing billions of public keys will call for a (mere) few hundreds of GBs of storage - a trivial requirement for any content distributor. Nevertheless, calling for storage proportional to $N$ certainly certainly cramps the scalability of the network (for example, making the use of identity based approaches impractical). Further, for T-BE schemes we have to take future scalability into account before we decide $N$.

On the other hand, for PKPSs, the storage required for external sources[7] is $PL \propto r\log(N')$, where $r$ is the maximum number of nodes that can be revoked in a single batch. As far as the KDC is concerned, revocation broadcasts by the KDC should reach all nodes. If the system actually has $N'$ nodes at some point in time, the KDC has to ensure outage probability $p_o \approx 1/N'$ to convey the secret to every node with a high probability. Similarly, the publishers (with group size $G$) only have to cater for $p_o \approx 1/G'$. Thus irrespective of the theoretical maximum network size $N \approx 2^{160}$ (to facilitate ID-based approaches) the KDS just has to cater only for the maximum number of users currently in the system.

*3) Privacy:* While in T-BE schemes the revoked devices have to be explicity specified in the revocation broadcast, recall that for PKPS-BE we only need to specify the indexes (and hash depths) of the $n_e$ secrets used in the broadcast (to encrypt the broadcast secret). Apart from protecting privacy of group membership information, we can also afford to use large IDs *without* adding to the bandwidth overheads.

*4) Storage for Secrets:* The advantages of PKPS-BE over schemes are achieved primarily by increasing the number of secrets assigned to every device (smart-card). Even for network size of $2^{60}$, T-BE schemes like the complete-subtree scheme require only storage for 60 secrets per node. However, a PKPS scheme requiring $n$ parallel deployments defined by parameters $(P_i, k_i, L_i), 1 \leq i \leq n$ calls for $\sum_{i=1}^{n} k_i$ secrets to be stored in each device (for example 800 secrets if $n = 4$ and each deployment has 200 secrets).

However, in any scenario calling for protection of multiple secrets, a very common approach (dating back to at least 1978 [20]) is to employ a single host master secret to encrypt all other secrets. Thus the smart-card $A$ belonging to a user Alice could store just one master secret $M_A$, and all other $\sum_{i=1}^{n} k_i$ decryption secrets assigned to $A$ could be encrypted using $M_A$ and stored outside the smart-card - for example in the hard-disk of Alice's desktop / laptop (or even a SD card that can be plugged into a PDA). Obviously, the storage complexity for the decryption secrets is not an issue. In other words we can increase the number of decryption secrets substantially to facilitate bandwidth efficient revocation (for example increasing $k$ by a factor $50 = e^{a-1}$ to reduce bandwidth requirement by a factor 5.

It is pertinent to point out that tree-based schemes substantially more efficient ($n_e/r \approx 1.25$) [7] have also been proposed which call for a storage complexity of $\log_2(N)^2/2$ keys (about 512 keys per device for $N = 2^{32}$, and 1800 keys for $N = 2^{60}$). However, such schemes extend less readily to BE by external sources.

### C. Pub-Sub Operation

A pub-sub system employing PKPS-BE will consist of a KDC who chooses a set of $n$ HARPS systems and public functions $F_i()$ and $f_i()$. Every participant in the system employs a smart-card, which uses and protects the $\sum_{i=1}^{n} k_i$ PKPS decryption secrets and group secrets on behalf of the participant. The smart card is plugged into a general purpose computer - for example desktop, laptop or PDA. Each smart card, associated with a user is assigned a 160-bit ID, based on the identity of the owner. For instance Alice has smart-card with ID $A$ with decryption secrets $\mathbb{S}_A$ and Bob has smart-card with ID $B$ and decryption secrets $\mathbb{S}_B$.

All users who desire to be publishers are also provided with a maximum of $\sum_{i=1}^{n} P_i L_i$ encryption secrets. To offset the cost involved generation and distribution of secrets by the KDC (the pub-sub operator) for this purpose, the KDC (or the pub-sub operator) could charge publishers a nominal fee to utilize the system for content distribution. Thus a publisher Alice also receives encryption secrets $\mathfrak{S}_A$. However operations involving encryption secrets are *not* performed by the smart card. The encryption secrets of Alice need not be protected from Alice. Thus operations involving $\mathfrak{S}_A$ can be performed by Alice's desktop computer. The secrets that are protected (hidden from the owner of the smart-card) by the smart-cards are 1) decryption secrets used for decrypting the broadcast secret, 2) the group secrets, and 3) the broadcast secrets (which are used to modify the group secrets).

Apart from providing encryption secrets to the publishers the tasks performed by the KDC include 1) proactive measure to identify compromised nodes, and revoke them from the system, and 2) renew secrets of the system periodically when a substantial number of nodes have been revoked. Specific approaches for identifying compromised devices for purposes of revocation, and renewal of the

[7] typically of the order of tens or hundreds of MB.

system over insecure networks (in which revoked devices will not be allowed to participate), is beyond the scope of this paper. Some possible approaches for renewal have however been investigated in [23] - [25].

*1) Establishing Group Secrets:* Consider a scenario where a publisher Alice, with smart-card $A$ inducts a member Bob with smart-card $B$. Apart from catering for BE, HARPS also facilitates establishment of shared secrets between any publisher (or any user who has encryption secrets) with any user with decryption secrets. For example, for using the system with $P = 204800, k = 200, L = 64$ for mutual authentication, Alice determines the $k$ indexes and the hash depths of the decryption secrets $\mathbb{S}_B$. Thus using $k$ of $PL$ encryption secrets $\mathfrak{S}^A$, Alice can encrypt a session secret $K_S$ that only Bob's smart-card can decrypt [18].

The publisher Alice chooses a group secret $K_{GA_0}$ and supplies the group secret to her smart-card $A$, which encrypts the secret $K_{GA_0}$ with the universal secret $U_i$. The secret $U_i(K_{GA_0})$ is now provided to the newly inducted member, over a channel secured using the established session secret $K_S$. Note that smart-cards revoked by the KDC (that do not have access to the secret $U_i$) cannot decrypt the group secret, and thus cannot become members of *any* group. By using the secret shared between the publisher and the subscriber to authenticate a commitment, hash-chain based approaches [21], [22] could be used for authenticating subsequent revocation broadcasts by the publisher.

*2) Revoking Users from Publishers Groups:* For revoking a set of $r$ users who have have access to the current group secret $K_{GA_j}$ of the publisher Alice, Alice chooses a new group secret $K_{GA_{j+1}}$, and encrypts it with the current universal revocation secret $U_i$ (by providing $K_{GA_{j+1}}$ to her smart-card $A$).

The secret $K_b = U_i(K_{GA_{j+1}})$ is broadcast by encrypting it with $n_e$ encryption secrets. Note that the primary complexity associated with PKPS-BE lies in the determining the indexes and hash depths of the $n_e$ secrets to use, for revoking a specific set of $r$ nodes (by evaluating the public functions). However, for purposes of creating the revocation broadcast, the only operation performed by the smart-card is encrypting the secret $K_{GA_{j+1}}$ to provide the publisher with the secret $U_i(K_{GA_{j+1}})$. All other operations are performed by Alice's desktop computer.

*Decryption of Group Secrets:* Depending on the nature of content distributed by the publisher, the revocation broadcast can be posted in the website of the publisher or distributed with the content. With PKPS-BE the distributor only indicates the indexes and the hash depths of the keys used.

At the other end, a subscriber Bob, using his desktop / PDA accesses the broadcast with $n_e$ encrypted versions of the broadcast secret, and a header indicating the indexes of the PKPS secrets. Bob's computer can evaluate public functions $F(B)$ (and $f()$) to determine an index of the secret that can be used by his smart-card $B$ to decrypt the broadcast secret $K_b$.

For instance, assume that 1) the broadcast includes $K_e(K_b)$ where $K_e = h({}^{63}K_{46} \parallel A)$, 2) $F(B)$ includes the index 46, and 3) $f(B, 46) = 42 < 63$. In other words, one of $B$'s decryption secrets $\mathbb{S}_B$ is ${}^{42}K_{46}$, which is stored in Bob's desktop computer as $M_B({}^{42}K_{46})$ (or encrypted with the master secret $M_B$ stored inside the the smart-card $B$). Thus Bob provides his smart-card with the values

$$[M_B({}^{42}K_{46}) \parallel K_e(K_b) \parallel 21 \parallel A] \qquad (14)$$

The sequence of operations to be performed by the smart-card to gain access to the group secret are as follows:

1) perform one decryption to determine ${}^{42}K_{46}$, and $21 = 63 - 42$ repeated hashes to evaluate ${}^{63}K_{46}$.
2) compute $K_e = h({}^{63}K_{46} \parallel A)$
3) decrypt $K_e(K_b)$ to determine $K_b$
4) decrypt $K_b$ with $U_i$ to determine new group secret $K_{GA_{j+1}}$
5) encrypt $K_{GA_{j+1}}$ with $M_B$, and
6) hand $M_B(K_{GA_{j+1}})$ back to Bob's desktop for storage

Thus the smart-card stores only the master secret $M_B$ and if necessary[8] the current universal secret $U_i$ (controlled by revocation broadcasts by KDC).

Any content distributed by the publisher Alice (with smart-card $A$), meant exclusively for her subscribers, is encrypted with a secret $K_C$. The secret $K_C$ is then encrypted with the group secret $K_{GA_{j+1}}$, and distributed along with the encrypted content. Thus Bob provides his smart-card $B$ with $M_B(K_{GA_{j+1}})$ and $K_{GA_{j+1}}(K_C)$. The smart-card performs two decryption operations to evaluate $K_C$. Depending on the nature of the specific application scenario and type of content, the secret $K_C$ may be handed over to untrusted (by the smart-card $B$) Bob, or only to a trusted DRM enabled device.

*3) Very-Low-Complexity Smart-Cards:* The scope of operations performed by smart-cards are restricted to a few symmetric cipher operations. The smart-cards are employed by publishers only for encrypting their group secret with the universal secret $U_i$. The subscribers employ it only for deciphering one of the $n_e$ encryptions of the broadcast secret. Thus even very low power processors (with a dedicated hardware block cipher) can be used, with practically no restrictions on the type of protection (in the form of shielding) that can be provided to protect the secrets and the operation of the processor in the smart-card. Apart from making it virtually impossible for attackers to successfully expose secrets from smart-cards, this can also result in inexpensive smart-cards. This one of the main reasons for avoiding the use of asymmetric cryptographic primitives inside the smart-card.

## VI. Conclusions

In this paper we proposed and investigated the performance of a family of BE schemes employing probabilistic

[8] While the universal secret can also be stored outside, it is necessary for decrypting all messages. Thus it may be more efficient to store the secret $U_i$ inside the smart-card.

key pre-distribution. Some of the very desirable properties of the proposed schemes include their ability to

1) permit broadcast by any source *without* the use of asymmetric cryptographic primitives;
2) cater for practically unlimited network sizes, and thus easily amenable to identity based approaches; and
3) conceal identities of revoked nodes - a useful feature in scenarios where privacy is a crucial.

Furthermore, PKPS-BE offers useful trade-offs between bandwidth, computation, and storage, which can make them useful for a wide variety of application scenarios. However in this paper we restricted ourselves to one (albeit with a broad scope) application scenario - secure content distribution under the publish-subscribe paradigm.

Apart from novel BE schemes, comprehensive analysis of their performance and bounds, and a broad application of PKPS-BE, the contributions of this paper also include a framework for BE applications depending on the relationships between the network size $N$ and group size $G$, and whether the devices are stateless of stateful. We argued that tree-based schemes are more suitable for stateless models with $N = G$, while PKPS-BE is ideally suited for $N >> G$ models with stateful devices.

## References

[1] M. Ramkumar, "On Broadcast Encryption with Random Key Pre-distribution Schemes," the Proceedings of the 1st Intl. Conf. Information Systems Security - ICISS 2005, Kolkata, India, December 2005.

[2] A. Fiat, M. Noar, "Broadcast Encryption," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **773**, pp 480–491, 1994.

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

[4] P.T. Eugster, P.A. Felber, R. Guerraoui, A-M. Kermarrec, "The Many Faces of Publish/Subscribe," Technical Report, URL citeseer.ist.psu.edu/649723.html.

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

[6] S. W. Smith, *Trusted Computing Platforms: Design and Applications,* Springer, New York, 2005.

[7] D. Noar, M. Noar, J. Lotspiech, "Revocation and Tracing Routines for Stateless Receivers," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **2139**, 2001.

[8] D. Halevy, A. Shamir, "The LSD Broadcast Encryption Scheme," Advances in Cryptology - CRYPTO 2002: 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002.

[9] C.K. Wong, M. Gouda, S. Lam, "Secure Group Communications using Key Graphs," Proceedings of SIGCOMM 98, pp 68–79, 1998.

[10] J. Anzai, N. Matsuzaki, T. Matsumoto, "A method for masked sharing of group keys (3)," IEICE Technical Report, ISEC99-38, 1999.

[11] J. Anzai, N. Matsuzaki, T. Matsumoto, "A quick group key distribution scheme with entity revocation," Proc. ASIACRYPT'99, LNCS1716, pp.333347, Springer-Verlag, 1999.

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

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

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

[15] 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.

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

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

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

[19] C. Wang, A. Carzaniga, D. Evans, A. L. Wolf, "Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems," Hawaii International Conference on System Sciences, January, 2002.

[20] S. M. Matyas, C. H. Meyer, "Generation, Distribution and Installation of Cryptographic Keys," IBM Systems Journal, **2**, pp 126 – 137, 1978.

[21] R.J. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas and R.M. Needham, " A New Family of Authentication Protocols," ACM Operating Systems Review, vol. 32, n. 4, pp. 9-20, October 1998, ACM Press.

[22] A. Perrig, R. Canetti, D. Song, D. Tygar, "Efficient and Secure Source Authentication for Multicast," in Network and Distributed System Security Symposium, NDSS '01, Feb. 2001.

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

[24] M. Ramkumar, "DOWN with Trusted Devices," IA Newsletter, **8** (4), Information Assurance Technology Analysis Center (IATAC), http://iac.dtic.mil/iatac/.

[25] M. Ramkumar , "Safe Renewal of a Random Key Predistribution Scheme for Trusted Devices," the 6th IEEE Information Assurance Workshop (The West Point Workshop), United States Military Academy, West Point, New York, June 2005.

**Mahalingam Ramkumar** obtained his PhD in electrical engineering from New Jersey Institute of Technology, Newark, NJ, in 2000.

He has been an Assistant Professor in the Department of Computer Science and Engineering, Mississippi State University, since August 2003. He was a co-founder and Chief Technology Officer of PixWave.com Inc., between March 2000 and August 2002, and Research Professor in the Department of Computer and Information Science, Polytechnic University, Brooklyn, NY, between September 2002 and August 2003.

Dr. Ramkumar has authored one book, 10 Journal papers, and over 50 Conference papers. His research interests include security under resource constraints, cryptography, ad hoc networks, data hiding and digital rights management.