Protocol Family for Optimal and Deterministic Symmetric Key Assignment

Yi-Hua E. Yang, Joseph D. Touch, *Senior Member, IEEE*

***Abstract*—Secure pairwise communications for a group of N nodes can require a large number of keys, $O(N^2)$ in the worst case. This work proposes a two-part algorithm for large-scale symmetric key pre-distribution: a key assignment algorithm deterministically assigning $O(\log_2 N)$ keys to each node, and a key discovery algorithm requiring $O(\log_q N)$ computations and inter-node messages to find the appropriate secure keys for a pair-wise communication. The lower bound of $O(\log N)$ complexity is achieved by novel applications of the Maximum-Distance Separable (MDS) codes. Compare to previous works, our scheme is deterministic, efficient, and has good security properties against collusion. Using $(n,k)_q$ Reed-Solomon codes as the MDS codes, we show by analysis that (1) in all practical cases, the lower bound of $c\log_2 N$ keys per node can be reached within 2 levels of recursions with a multiplicative constant c less than 8; (2) by increasing cost to $k^2 O\left(\log_2^2 N\right)$, collusion of up to $(\log_2 N)/2$ nodes can be completely prevented; (3) channel resilience against r-collusion is roughly $1 - R^{d_{\min}}$, with $R = 1-(1-1/q)^{2r}$ and $d_{\min} = n-k+1$.**

***Index Terms*—Collusion resilience, Maximum-Distance Separable code, Reed-Solomon code, symmetric key assignment, symmetric key pre-distribution.**

I. Introduction

DIFFERENT schemes of symmetric key pre-distribution have been proposed and studied in the context of conference communication [4], wireless sensor networks [5], [6], [7], [8], [12], and Internet broadcast security [9]. There are two main stages for a key pre-distribution process: (1) the *key assignment phase*, where each node is assigned a set of secret (symmetric) keys, and (2) the *key discovery phase* (also called the key agreement phase), where two nodes discover their shared keys for secure pairwise communication. To achieve pairwise security, it is necessary and sufficient to generate N distributed key sets, one for each node, that possesses both *sharing* and *exclusion* properties [1] among one another. We call this the *key assignment problem* of size N, which must be solved during the beginning of the key assignment phase. After the key assignment, any two nodes wishing to communicate with each other securely need to perform a secure key discovery per channel establishment. Subject to the properties of the assigned key sets, the two nodes will exchange some clear-text information to agree upon the shared secret keys to use.

The key assignment stage is associated a *local* and a *glocal key assignment cost*, defined to be the number of secret keys required for each of the N nodes (local) and for all N nodes in total (global), respectively. The key discovery stage is associated with additional communicational and computational complexities. The simplest solution, assigning $N-1$ keys to every node, one for communication with every other node, requires a local cost of $O(N)$ and a global cost of $O\left(N^2/2\right)$, and a constant key discovery (just the two node IDs). A family of more sophisticated, t-composite key distribution [5] schemes protect each communication channel by $t>1$ keys; they can achieve lower key assignment costs with a more expensive key discovery.

A t-composite key pre-distribution scheme can be *deterministic* or *probabilistic*. In a deterministic scheme, key sets are assigned in a non-probabilistic manner, which guarantees some values of node connectivity and channel security. In a probabilistic scheme, key sets assigned to each node are drawn randomly from a key pool according to some preset parameters, inducing probabilistic channel security and/or node connectivity. A deterministic scheme usually has higher key assignment costs, whereas a probabilistic scheme has more complex key discovery.

In this paper, we propose a novel application of the Maximum-Distance Separable (MDS) codes to the key assignment problem. By doing so, we obtain a t-composite key pre-distribution scheme that is deterministic, achieves optimal $O(\log N)$ key assignments, has low $O\left(\log^2 N\right)$ computational complexity, and forms a fully connected graph among all nodes.

II. Related Works

In [3], Blom describes a Symmetric Key Generation System that, utilizaing an MDS code, sends only $r\log(N)$ bits of secret information to each node to guarantee pairwise security against r-collusion. There are however several problems with this system regarding its practicality. First, to assign keys to N nodes against r-collusion, the key set generation requires the use of an $(N,r)_q$ MDS code, and the key discovery requires the multiplication of two size-r and size-$(N-r)$ vectors over a q-ary alphabet with $q \geq N$. The computational complexity thus increases in the order of $O\left(rN^2\right)$, making the system impractical when N is more than a few thousands and r is a few tens. Second, the system has a "hard" collusion resilience; that is, it is 100% secure against collusion of less than r nodes, but totally broken when r or more nodes collude.

Manuscript received Nov.17, 2007. This work was supported by the National Science Foundation under Grant SCI-0225642. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do no necessarily reflect the views of the NSF.

Y. E. Yang is with University of Southern California / Infornation Sciences Institute, Marina del Rey, CA 90292, USA (phone: 310-448-8858, e-mail: yeyang@isi.edu)

J. D. Touch is with University of Southern California / Infornation Sciences Institute, Marina del Rey, CA 90292, USA (e-mail: touch@isi.edu)

Blundo *et al.* in [4] generalize Blom's work to securing dynamic conferences. Their approach reaches the lower bound of information that must be held by each node in the more general multi-way conferencing; instead of using an MDS code, they use a t-variable, k-degree symmetric polynomial to establish a secure conference among t users against k-collusion. The scheme solves a more generalized problem, but has the same high complexity as Blom's in the reduced 2-conference cases, requiring the evaluation of a degree r polynomial over an alphabet of size $q > N$. In [12], Mohaisen and Nyang add a grid-based hierarchy on top of Blundo's approach, allowing the system to scale to larger number of nodes with most communications occur locally. The memory requirement in this hierarchical scheme, suppressing the lower-order terms, is roughly $O\left(nk^2\right)$, where n is the depth of the hierarchy and k^2 the size of the leaf group.

A non-MDS approach is taken by Delgosha *et al.* in [6], where nodes are put conceptually onto the grid of a hypercube and securely communicate only with adjacent nodes. Their solution trades full pairwise connectivity off for a smaller key assignment cost and better collusion resilience. A similar, hypercube grid-based scheme is proposed by Aiyer *et al.* in [1]; instead of assigning keys to a node by its hypercube position, they assign keys by the node's coordinate projection onto each of the 2-dimensional planes composed of any 2 axes of the hypercube; the lowest key assignment cost obtained by their scheme is $4\log_2^2 N$. In the same paper they also prove non-constructively the existence of the lower-bound solution of $O\left(\log_2 N\right)$ keys per node.

Of the probabilistic methods, Du *et al.* extends Blom's scheme such that each node is assigned keys from a randomly chosen t out of w key spaces [8]. Although in this scheme an arbitrary pair of nodes is not guaranteed to find a set of shared keys, if they do they can establish a secure channel with better collusion resilience. Other probabilistic schemes, like [5], [7], [9], and [13], etc., follow the same basic idea that lets each node randomly select K distinct keys out of a pool of P total keys. All these schemes achieve pairwise connectivity and security with a pre-configured but less-than-one probability. Their focus are on the higher-layer protocols that utilize the probabilistic schemes to facilitate efficient key deployment and discovery. In particular, [13] applies a one-way hash function to multiple degrees on each of the P root keys in a pool, and only assigns K hashed keys (with different hash degrees) to the nodes. It has the same connectivity (two nodes can always hash forward to find a common key) as a non-hashed solution, but better collusion resilience (*e.g.*, adversary assigned a key with high hashing degree cannot break communication channels protected by keys with lower hash degree).

III. MDS Key Sets Distribution

Here are the assumptions we use for the key pre-distribution problem:

1) We secretly generated M symmetric keys to be distributed to the N nodes. M is the global key assignment cost.
2) A key distribution center has a secure channel to each of the N nodes to send the symmetric keys to the node.
3) Each node is associated with an integer *node ID* to identify itself uniquely and universally within the network.

The key distribution scheme consists of a *key assignment algorithm* and a *key discovery algorithm*. Both algorithms utilize the properties of the MDS codes, reviewed in the following subsection.

A. Review of MDS Codes

Recall that an $(n,k)_q$ MDS code, C, over a finite alphabet F of size q is a collection of q^k codewords of length n satisfying the condition that no two codewords in C agree in as many as k coordinate positions. An MDS code can be either linear or non-linear. In the case of a linear MDS code, F consists of symbols of the Galois field GF(q), and the code C meets the *Singleton bound* of minimum (Hamming) distance, $d_{\min} = n - k + 1$. The most commonly used linear MDS codes are the Reed-Solomon codes (RSC).

The following theorem contains a list of properties of the MDS codes that are of our concern.

Theorem 1: Let a C be an $(n,k)_q$ Maximum-Distance Separable code. Below is a list of properties of C relevant to our discussion [11], [14]:

1) For any $t \leq k$ coordinate positions of C, we can find exactly q^{k-t} codewords in C with these t coordinate symbols fixed to any of the q^t values.
2) Any fixed k coordinates in C carry the full information of k symbols. An equivalent description of this is that if a set of codewords are uniformly and randomly selected from C, their symbol values in any fixed k coordinates are also uniformly random.
3) A *shortened* MDS code is also MDS. Fix any coordinate j of C and specify an arbitrary value for it, the set of the resulting codewords, removed of the j-th, coordinate is an $(n-1,k-1)_q$ MDS code.
4) A *punctured* MDS code is also MDS. Removed of any fixed coordinate of C results in an $(n-k,k)_q$ MDS code.
5) If C is linear, then it can be constructed over a finite field of GF(q), $q \geq n-1$, using a generator matrix of dimension $k \times n$ with entries from GF(q). In the case of Reed-Solomon codes, a generator polynomial of degree $n-k$ with coefficients in GF(q) can be used.

Although most of the steps and proofs in our study do not require linearity, in this paper we consider only linear MDS codes, most prominently the Reed-Solomon codes. This is because little is known about the availability of practical nonlinear MDS codes; besides, there are already efficient encoders for the Reed-Solomon codes. We note that our scheme requires only the encoding of the codes, but not the more complex decoding process.

B. The Key Assignment Algorithm

The key assignment algorithm solves the *key assignment problem* defined below.

Algorithm 1 The Key Assignment Algorithm

```
BEGIN
  1) For each node $A$, find its key-assign ID $[\alpha]_{(n,k)}$.
  2) For each $i$-th symbol coordinate of the key-assign ID, $i = 0,\ 1,\ \ldots,\ n-1$, assign the following set of symbol keys to node $A$,
         $s_i(\alpha_i) \equiv \bigcup_{\beta \in \mathtt{GF}(q)} s_i(\alpha_i, \beta)$ ,
     such that the following two properties hold:
       a) Sharing. For all $u$ and $v$ from $\mathtt{GF}(q)$, $|s_i(u,v)| > \emptyset$ and $s_i(u,v) = s_i(v,u)$;
       b) Exclusion. For all $u$ and $v$ and any other $x$ from $\mathtt{GF}(q)$, $s_i(u,v) \nsubseteq s_i(x)$.
  3) Repeat step 1 and 2 for all nodes.
END
```

Definition 1: The *key assignment problem* of size N. Given N nodes numbered $0, 1, \ldots, N-1$, for any pair of nodes $\{i, j\}$ and any other node k, assign sets of symmetric keys $s(i)$, $s(j)$, and $s(k)$ to nodes i, j, and k, respectively, satisfying the following properties:

1) For all i, j, k, there exists some constant t such that
$$t \geq \{|s(i)|, |s(j)|, |s(k)|\} > 0 .$$
The key assignment is said to have a local cost of t keys per node.
2) Let $s(i,j) = s(i) \cap s(j)$; then
   a) $|s(i,j)| > 0$ (sharing)[1].
   b) $s(i,j) \nsubseteq s(k)$ (exclusion)[1].

In contrast to the grid-oriented key assignment method used in [1] and [6] or the value-oriented method used in [3], we adopt a *coordinate*-based key assignment scheme (similar to [9]). Instead of directly using the node IDs, however, we represent each node ID as a vector of k coordinates and derive its key-assign ID using the following definition.

Definition 2: The *key-assign ID* (a q-ary vector),
$$[\alpha]_{(n,k)} \equiv [\alpha_1,\ \alpha_2\ , \ldots,\ \alpha_{n-1}] \ ,$$
of a node A is the value of its node ID (also a q-ary vector),
$$[a]_k \equiv [a_0,\ a_1,\ \ldots,\ a_{k-1}] \ ,$$
encoded by an $(n,k)_q$ MDS code with $n \geq 2k-1$.
Note that there is a one-to-one mapping between the node ID and the key-assign ID. Also note that, by definition, $n \geq 2k-1$. As is shown later in section III-D, this simple constraint guarantees both sharing and exclusion properties among the assigned key sets.

Horizontally, Algorithm 1 divides a size-N key assignment problem into $n \geq 2k-1$ units of size-q subproblems (one per key-assign ID symbol coordinate). Step 2a and 2b asserts the sharing and exclusion properties among the *symbol key sets* in each subproblem. In the cases where q is fairly small (*e.g.*, 8 or less), we can assign q different keys to each symbol value. That is, the simplest solution for the subproblem at the i-the symbol can be constructed as follows:

1) For any pair of values u and v in $\mathtt{GF}(q)$, let $s_i(u,v)$ be a single key.
2) Assign $s_i(u) = \{s_i(u,0), s_i(u,1), \ldots, s_i(u,q-1)\}$ to the symbol key set corresponding to a symbol value u.

The simple assignment assigns q out of $q^2/2$ keys to each symbol coordinate of a key-assign ID, resulting in a total of $(2k-1)q$ keys per node, out of $(2k-1)q^2/2$ keys for all nodes. It has a local key assignment cost of $O\left(2q\log_q N\right)$ and a global cost of $O\left(q^2 \log_q N\right)$; this is better than the best previous result $O(\log_2^2 N)$ [1], although still higher than the optimal $O(\log N)$, especially when N is large and q (a non-arbitrary design constant) is in the order of $O(\log N)$.

C. Recursive and Mixed Key Assignments

Fortunately, we can take advantage of the recursive nature of Algorithm 1 to further reduce the key assignment cost of every size-q subproblems.

More specifically, we treat each of the n symbols in the key-assign ID as a sub-node ID, which can be represented by a vector of k_1 elements over $\mathtt{GF}(q_1)$ with $q_1^{k_1} \geq q$. We then apply Algorithm 1 to every sub-node ID, choosing a proper (n_1, k_1) MDS code with $n_1 \geq 2k_1 - 1$ to derive a solution for the corresponding size-q subproblem. These subproblem solutions will have local and global key assignment costs $O\left(2q_1 \log_{q_1} q\right)$ and $O\left(q_1^2 \log_{q_1} q\right)$, respectively. With $n \approx 2\log_q N$, we get local and global costs for the full problem $O\left(4q_1 \log_{q_1} N\right)$ and $O\left(2q_1^2 \log_{q_1} N\right)$, respectively.

If q_1 is still large, we can repeat the above process recursively. With j-level recursion we get local cost $O\left(2^{j+1} q_j \log_{q_j} N\right)$ and global cost $O\left(2^j q_j^2 \log_{q_j} N\right)$. Although every additional level of recursion adds a multiplicative constant 2 to the costs of the full problem, the sequence of $N,\ q,\ q_1,\ \ldots$, however, can decrease super-exponentially ($q_{j+1} \approx \log_{q_j} q_j$).

We notice that Algorithm 1 poses no other requirement on the symbol key sets for the subproblems as long as they constitute *one* solution to the key assignment problem of size q. The subproblems can use the simplest $O(q)$ solution, another recursion of the algorithm, or a totally different key assignment solution (*e.g.*, from [1] or [3]). Thus by design, Algorithm 1 can be used to assemble a few (at least $2k-1$) existing key assignment solutions of size q to construct a key assignment solution of size q^k. Inversely, it can be used to dissect a large key assignment problem into multiple smaller ones, each to be solved by a different authority, for example, for security reasons.

The ability to mix different key assignment solutions in the lower level is more powerful than just recursion. For example, a major problem of the Symmetric Key Generation System in [3] is the use of high-degree MDS codes when the problem size (N) is large. However, such a system would work well for the lower (recursed) level in our scheme, where the size of the subproblem is a much smaller q. If we make $n \geq q$, we can even reuse the $(n,k)_q$ MDS code of the full problem for all the subproblems being solved by SKGS.

Algorithm 2 The Key Discovery Algorithm

```
BEGIN
  1) Nodes A and B exchange their key-assign IDs ($[\alpha]_{(n,k)}$
     and $[\beta]_{(n,k)}$, respectively) with each other.
  2) Nodes A and B initialize their respective shared key sets,
     $S_A$ and $S_B$, to the empty set.
  3) For each (key-assign ID) symbol coordinate $i = 0, 1, \ldots, n-1$,
     A adds the set of symbol keys $s_i(\alpha_i, \beta_i)$ to $S_A$ and
     B adds the set of symbol keys $s_i(\beta_i, \alpha_i)$ to $S_B$.
  4) Repeat step 3 for every key-assign ID coordinate.
END
```

D. The Key Discovery Algorithm

The key discovery algorithm is used to find the common keys assigned to any two of the N nodes (A and B) for secure pairwise communication.

We note that the first step in this algorithm do *not* require additional authentication; the ability to construct the shared key set *S* in the end will implicitly authenticate the exchanged key-assign ID. Also, Algorithm 2 works as-is when Algorithm 1 is applied recursively; the only difference is in the construction of $s_i(\alpha_i, \beta_i)$ and $s_i(\beta_i, \alpha_i)$, where under recursion these two (identical) sets must also be derived recursively by both *A* and *B*.

For Algorithm 2 to establish a secure channel, we need $2\log_q N$ symbol value comparisons to find the shared key set. The message exchange is just the two key-assign ID vectors of length n, making the communication cost also in $O\left(\log_q N\right)$.

E. Secure Pairwise Communication

Lemma 1: Assume nodes *A*, *B*, and *X* have key-assign ID vectors $[\alpha]_{(n,k)}$, $[\beta]_{(n,k)}$, and $[\chi]_{(n,k)}$, respectively, and are assigned symmetric keys according to Algorithm 1. If the set of shared keys *S* constructed by *A* and *B* following Algorithm 2 is wholly assigned to *X*, then $d_{\alpha\chi} + d_{\beta\chi} = d_{\alpha\beta}$.

Proof: Let S consists of n subsets of keys, $s_0, s_1, \ldots, s_{n-1}$, one for each coordinate of the key-assign ID. For any i-th coordinate, step 3 of Algorithm 2 ensures that $s_i \equiv s_i(\alpha_i, \beta_i) = s_i(\beta_i, \alpha_i)$. According to step 2b of Algorithm 1, for *X* to receive all the keys in s_i, χ_i must be equal to either α_i or β_i. Applying this argument to every coordinate $i = 0, 1, \ldots, n-1$, we have $d_{\alpha\chi} + d_{\beta\chi} = d_{\alpha\beta}$. ■
Note that the proof does not depend on *how* the key set s_i for the i-th symbol coordinate is derived, as long as the q sets of s_i, $i = 0, 1, \ldots, n-1$, form a solution to the key assignment problem of size q. Therefore Lemma 1 holds even when Algorithm 1 is applied recursively.

Theorem 2: The key assignment in Algorithm 1 and key discovery in Algorithm 2 allow a secure channel to be established by a unique set of shared keys between any pair of the N nodes.

Proof: The Hamming distance between any two codewords of an (n, k) MDS code is at most n (total number of symbols) and at least $n - k + 1$, *i.e.*, $n \geq d_{\alpha\beta} \geq n - k + 1$. Assume a third node *X* is necessarily assigned all the keys shared by *A* and *B*. According to Lemma 1, this implies $n \geq d_{\alpha\beta} = d_{\alpha\chi} + d_{\beta\chi}$. On the other hand, $d_{\alpha\chi} + d_{\beta\chi} \geq 2(n - k + 1) = n + 1 + n - (2k - 1) > n$ (by design, $n \geq 2k - 1$). By contradiction such a node *X* does not exist. ■

As a side note, from proves above we can see that the MDS codes are not the only type of codes that can be used in our scheme. In general, as long as an (n, k) code has maximum distance $d_{\max}$ and minimum distance $d_{\min}$ such that $d_{\max} < 2d_{\min}$, it can be used by Algorithm 1 to transform the node ID to the key-assign ID. The (n, k) MDS codes with $n \geq 2k - 1$ is only a special case where $d_{\max} = n$ (implicitly) and $d_{\min} = n - k + 1$.

IV. Collusion Analysis

While no single third node in our scheme possesses all the keys used in any pair-wise communication, two or more nodes can *collude* to pool their assigned keys together, allowing *all* colluding nodes to break a secure channel between *some other* pairs of nodes in the system. The process of r nodes pooling their keys together for such purpose is called r-collusion.

A. Collusion Prevention

Assume two nodes *A* and *B* share the key set $s_i\left(\alpha_i, \beta_i\right)$ on the i-th coordinate of the key-assign ID. It's obvious that this key set is compromised (*covered*) by a third node *X* if and only if χ_i equals either α_i or β_i. Extending this argument to $r \geq 2$ colluding nodes results in the following lemma.

Lemma 2: In general, the secure channel established by the shared keys between two nodes *A* and *B* is compromised by *r* colluding nodes, $X^{(j)}$, $j = 0, 1, \ldots, r-1$, if and only if some $\chi_i^{(j)}$ equals either α_i or β_i (or both if $\alpha_i = \beta_i$) for *every* key-assign ID coordinate $i = 0, 1, \ldots, n-1$.

Proof: Similar to what is described in the paragraph above. ■

Theorem 3: The communication channel established by the shared keys between any two nodes is *resilient against* r-*collusion* (*i.e.*, r-collusion is *prevented*) if the key assignment is performed with an $(n, k)_q$ MDS code where $n > 2r\left(k - 1\right)$.

Proof: The minimum distance of an MDS code, $d_{\min} = n - k + 1$, implies that the maximum number of equal symbols between any two codewords is $e_{\max} = n - d_{\min} = k - 1$. Thus outside of a pair of nodes *A* and *B*, any third node can have at most $e_{\max}$ symbols equal to those of *A* and another $e_{\max}$ to those of *B*, and r colluding nodes can pool together at most $2re_{\max} = 2r\left(k - 1\right)$ symbol key sets used between *A* and *B*. As long as (the number of symbol key sets shared by *A* and *B*) $n > 2r\left(k - 1\right)$, the channel between *A* and *B* is secure against r-collusion. ■
Increasing the length of the MDS code to prevent collusion is rather expensive. For each increase in r, one must pay $2\left(k - 1\right) \approx 2O\left(\log_q N\right)$ extra key assignment cost. For large r this may not even be possible, since the existence of the MDS code is mostly bounded by $n \leq q + k - 3$, although the exact bound is still an open question [2]. Let $k \approx q/\log_2 N$ and use $n \approx q$, the key assignment cost

becomes $qn \approx k^2 \log_2^2 N$ and can prevent collusion of up to $(\log_2 N)/2$ nodes.

As an example, for $N = 2^{32}$, the key assignment using an RS code of $(n,k)_q = (120,5)_{125}$ can prevent collusion of up to $120/8 = 15$ nodes with ~1800 keys per node. The key assignment cost in this example is comparable to the best result in [1]; our scheme, however, has the benefit of absolute security against up to 15 colluding nodes.

B. Collusion Resilience

Definition 3: The r-collusion resilience, or *r-resilience*, of a key pre-distribution scheme is the probability of a communication channel being secure when established outside of a random set of r colluding nodes. This probability can be calculated as the ratio of secure over total number of pairwise communication channels under the r-collusion.

To calculate the collusion resilience of our key assignment scheme, we first introduce the concept of *pair-group of r-resilience*, whose size equals the number of resilient channels under r-collusion. Then we estimate the pair-group size using the number of *collusion-free symbol values* under the r-collusion.

1) Pair-group of r-resilience: For any pair of nodes *A* and *B* to establish a secure channel against the 2-collusion between nodes *X* and *Y*, there must be at least one key-assign ID coordinate at which both *A* and *B* have values different from those that both *X* and *Y* have. We say all such $\{A,B\}$ forms a *pair-group of 2-resilience* against *X* and *Y*. The following definition extends this pair-group notion to r-resilience.

Definition 4: The *pair-group of r-resilience* against a random set of r colluding nodes $X^{(j)}$, $j = 1,2,\ldots,r$, consists of all pairs of nodes $\{A,B\}$ such that, at *some* i-th symbol of the key-assign ID, $\alpha_i \neq \chi_i^{(j)}$ and $\beta_i \neq \chi_i^{(j)}$ for all j.

In other words, the pair-group of r-resilience consists of all pairs of nodes which can still establish secure channels amid the r-collusion. Note the pair-group definition does not require any MDS property nor a solution to the key assignment problem. To estimate the size of the pair-group we need to following definition and lemma.

Definition 5: Assume a key assignment is under r-collusion. A *collusion-free symbol value* at an arbitrary coordinate of the key-assign ID is a symbol value not held by any of the r colluding nodes at that coordinate.

Lemma 3: Assume a key assignment under r-collusion is performed with an $(n,k)_q$ MDS code. Let $G(n,k)$ be the size of the pair-group of r-resilience. Then the following recursive equation holds:

$$G(n,k) = u^2 \times q^{2(k-1)} + \left(1 - \frac{u^2}{q^2}\right) \times G(n-1,k), \quad (1)$$

where u is the number of collusion-free symbol values at the (arbitrary) coordinate where the recursion takes place.

Proof: Without loss of generality, assume the recursion takes place at the first coordinate (*i.e.*, index 0) of the key-assign ID. Every pair of nodes resilient to the r-collusion falls in either one of the two following cases:

Case 1: The channel is protected by the first symbol coordinate. Both communicating nodes hold collusion-free symbol values at the first symbol coordinate. There are u^2 (out of q^2) such value pairs in total. According to property 1 of Theorem 1, each such value pair can make $q^{k-1} \times q^{k-1} = q^{2(k-1)}$ pairs of codewords, giving the first term of (1).

Case 2. The channel is not *protected by the first, but the rest $n-1$ symbol coordinates.* Ignoring the first symbol coordinate of the key-assign ID, the rest $n-1$ symbols constitute an $(n-1,k)_q$ MDS code (property 3 of Theorem 1), where every u^2 out of q^2 codewords were already counted in Case 1. There are thus $1 - u^2/q^2$ times $G(n-1,k)$ additional codeword pairs resilient to the r-collusion, giving the second term of (1). ∎

The recursion in (1) reduces the pair-group size of an $(n,k)_q$ MDS code to that of an $(n-1,k)_q$ MDS code, assuming we know the value of u at the (arbitrary) first coordinate. After $n-k$ steps the recursion will leave only k symbols in a trivial $(k,k)_q$ MDS code (all the q-ary vectors of length k), where we calculate $G(k) \equiv G(k,k)$ in the following lemma.

Lemma 4: Assume a key assignment performed with the set of q-ary vectors of length k is under r-collusion. Let $G(k)$ be the size of the pair-group of r-resilience and u the number of collusion-free symbol values at an arbitrary coordinate. Then,

$$G(k) = u^2 \times q^{2(k-1)} + \left(1 - \frac{u^2}{q^2}\right) \times G(k-1). \quad (2)$$

Obviously, $G(0) = 0$.

Proof: We have the same two cases and the same number of collusion-resilient pairwise communicating channels as in Lemma 3. Here, however, the “removal” of any coordinate also reduces k by one. This reduction goes on until no symbol is left to protect against r-collusion and $G(0,0) = 0$. ∎

2) Number of Collusion-free Symbol Values: The main difficulty in expanding the recursions in (1) and (2) lies in the unknown number of collusion-free symbol values against the r-collusion, *i.e.*, the value of u. Clearly, the actual value of u depends on the particular key-assign ID patterns of the r colluding nodes. However, since the symbol values of an MDS code at any coordinate is randomly and uniformly distributed if the codewords are chosen the same way (property 2 of Theorem 1), we can *estimate* $G(n,k)$ and $G(k)$ by using an expected value of u.

Assume the r colluding nodes collectively hold w unique symbol values, $1 \leq w \leq r$, at some key-assign ID coordinate where the recursion takes place. The number of collusion-free symbol values is thus $u = q - w$, $q - r \leq u \leq q - 1$. In addition, $1 \leq w \leq n$ and thus $0 \leq u \leq n-1$. We define $\hat{w} \equiv \hat{w}_{q,r}$ as the expected number of unique values from r random picks out of q values with replacement (*i.e.*, let the r colluding nodes each pick one value out of the q symbol values at the symbol coordinate):

$$\hat{w}_{q,r} = q - q\left(1 - \frac{1}{q}\right)^r \quad , \quad \hat{u} = q - \hat{w} = q\left(1 - \frac{1}{q}\right)^r (3)$$

Equation (3) above is derived from the following recursion:

$$\begin{aligned}\hat{w}_{q,r} &= \frac{\hat{w}_{q,r-1}}{q}\hat{w}_{q,r-1} + \frac{q-\hat{w}_{q,r-1}}{q}\left(1+\hat{w}_{q,r-1}\right) \\ &= 1+\left(1-\frac{1}{q}\right)\hat{w}_{q,r-1} = \sum_{i=0}^{i=r-1}\left(1-\frac{1}{q}\right)^i .\end{aligned}$$

Using the expected value of $\hat{u}$ in (3), we define the *collusion ratio* at the symbol position to be the number of symbol value pairs compromised by the r-collusion,

$$R \equiv 1 - \frac{\hat{u}^2}{q^2} = 1 - \left(1-\frac{1}{q}\right)^{2r} . \quad (4)$$

The recursions in (1) and (2) can then be estimated as follows:

$$\begin{aligned} G(n,k) &\approx \hat{u}^2 q^{2(k-1)}\left[1+R+\ldots+R^{n-k-1}\right] \\ &+ R^{n-k}G(k) \quad (5)\end{aligned}$$

$$G(k) \approx \hat{u}^2 q^{2(k-1)}\left[1+\frac{R}{q^2}+\ldots+\left(\frac{R}{q^2}\right)^{k-1}\right] . \quad (6)$$

Merging (5) and (6) and adding up the geometric sums with $d_{\min} = n-k+1$, we get

$$\begin{aligned} G(n,k) &\approx q^{2k}\left(1-R^{d_{\min}}\right) + \hat{u}^2 R\left[\frac{q^{2(k-1)}-R^{k-1}}{q^2-R}\right] \\ &\approx q^{2k}\left[1-R^{d_{\min}}+\frac{R\,(1-R)}{q^2}\right] . \quad (7)\end{aligned}$$

The last approximation in (7) comes from $\hat{u}^2 = q^2(1-R)$ in (4) and the fact that $R < 1 \ll q^2$.

From above we get the following theorem.

Theorem 4: The r-resilience in Definition 4 of an MDS key distribution scheme using an $(n,k)_q$ MDS code for the key assignment is roughly

$$1 - R^{d_{\min}} + \frac{R\,(1-R)}{q^2} , \quad (8)$$

where $d_{\min} = n-k+1$ and $R = 1-(1-1/q)^{2r}$.

Proof: Dividing (7) by the total number of pairwise channels, $\left(q^k\right)^2 = q^{2k}$. ∎

To get higher r-resilience in our key distribution scheme we could use an MDS code with either higher $d_{\min}$ or a lower R (*i.e.*, higher q). The dependencies of resilience on n (codeword length) and k (information size) are carried by both $d_{\min}$ and R implicitly (a higher value of q implies a higher maximum value of n, as described in the end of section IV-A).

Fig. 1 plots r-resilience of several key assignments for 4096 nodes versus the number of colluding nodes. With larger q, resilience drops much more slowly against increasing r; the price however is the higher key assignment cost and the MDS code complexity.

For q = 16 and 64, resilience to r-collusion with different values of $d_{\min}$ are also examined. From the graph we see that doubling $d_{\min}$ has the effect of right-shifting the curve against r by roughly $q/4$; in other words, by assigning 50% more keys to each node, the key distribution can withstand collusion among $q/4$ more nodes with the same probability.

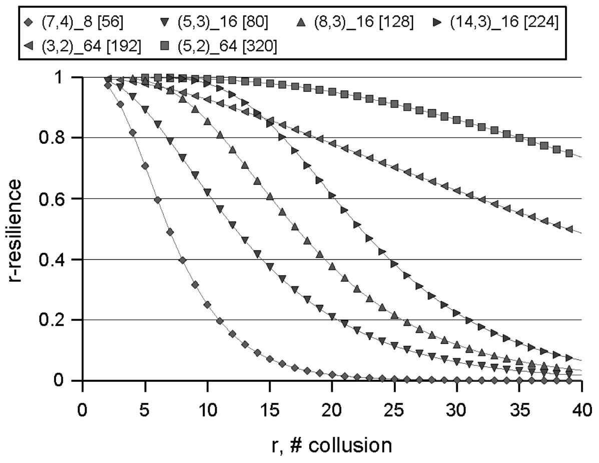

Figure 1. Graph of r-resilience of key assignments for 4096 nodes. The label convention is *(n,k)_q* [# keys].

Table I
SMALL-SCALE KEY ASSIGNMENT EXAMPLES

Size N	$\log_2 N$	RS code	# keys	Cost
16	4	$(3,2)_4$	12	$3.0 \times \log_2 N$
25	4.6	$(3,2)_5$	15	$3.2 \times \log_2 N$
64	6	$(5,3)_4$	20	$3.3 \times \log_2 N$
343	8.4	$(5,3)_7$	35	$4.2 \times \log_2 N$
4096	12	$(7,4)_8$	56	$4.7 \times \log_2 N$
32k	15	$(9,5)_8$	72	$4.8 \times \log_2 N$

V. REED-SOLOMON CODES EXAMPLES

In this section we look at some examples of our scheme implemented by the Reed-Solomon (RS) codes. RS codes are a special family of MDS codes; with RS codes, q (order of the Galois Field) must be in the form of p^h where p is prime and h is a natural number. It is conceivable to find a different MDS code with less restriction, which is out of the scope of this study.

A. Non-recursive Examples

Table I lists a few small-scale key assignment examples using Algorithm 1. As can be seen from the table, the constant c in the local cost of $c\log_2 N$ increases slowly with larger q. Since $q^k = N$ and q increases exponentially slower than $\log_2 N$, the value of c is well below 10 in all cases.

B. Recursive Examples

To assign a set of symmetric keys to every IPv4 address so that any pair of the 2^{32} hosts can establish a secure channel between them, we could apply Algorithm 1 first to the full problem with a $(15,8)_{16}$ RS code, then to every size-16 subproblem with a $(3,2)_4$ RS code, assigning 12 keys (3×4) to the node on each subproblem. Without recursion, a host is assigned $15\times 16 = 240$ keys; with the two-level recursion, the host is assigned $15\times 12 = 180$ keys, or a local key assignment cost of $5.6 \times \log_2 N$.

In a practically limiting case, assume the problem size is 2^{256} (about one node per atomic particle in the universe). The node ID can be represented as 37 symbols in $\mathtt{GF}\,(125)$.

Table II
TWO-LEVEL KEY ASSIGNMENTS USING RS CODES

Size N	Level 1	Level 2	# keys	Cost
2^{20}	$(9,5)_{16}$	$(3,2)_4$	108	$5.4 \times \log_2 N$
2^{32}	$(15,8)_{16}$	$(3,2)_4$	180	$5.6 \times \log_2 N$
2^{128}	$(43,22)_{64}$	$(5,3)_4$	860	$6.5 \times \log_2 N$
2^{256}	$(73,37)_{125}$	$(5,3)_5$	1825	$7.1 \times \log_2 N$

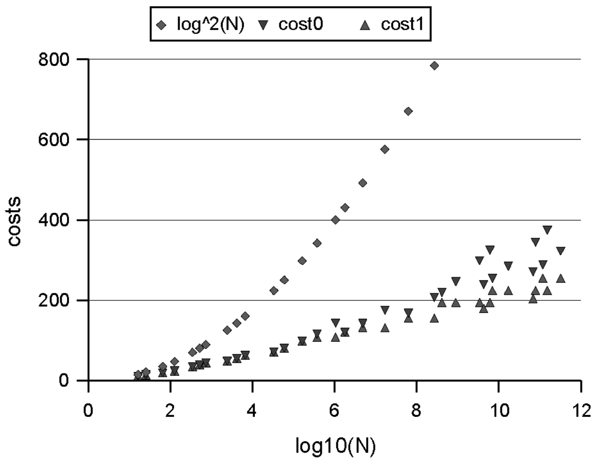

Figure 2. Comparisons of key assignment costs between non-recursive ("cost0") and recursive ("cost1") applications of Algorithm 1.

We could apply Algorithm 1 first to the full problem with a punctured $(73,37)_{125}$ RS code, then to each of the 73 subproblems with a $(5,3)_5$ RS code . The two-level recursion assigns $73 \times 5 \times 5 = 1825$ keys per node, or a local key assignment cost of $7.1 \times \log_2 N$.

Table II lists a few more recursion examples of Algorithm 1. In all practical cases, the multiplicative constant c in the local cost $c \log_2 N$ is always less than 8. In [1], the lower bound key assignment cost was proved to be $9 \times \log_2 N$.

C. Recursion on Key Assignment Cost

Fig. 2 compares non-recursive (label "cost0") and recursive (label "cost1") local key assignment costs of different problem sizes. The $\log_2^2 N$ line is provided to compare our solutions to the lowest costs in [1].

A major advantage of recursion is that it allows better collusion resilience (larger q or higher d_{min}) with the same range of key assignment costs. For example, for 15k nodes, a recursive key assignment using a $(24,3)_{25}$ RS code assigns 360 (a bit less than $2 \times \log_2^2 N$) keys per node, can completely prevent all 6-collusions, and have > 99% resilience to any random 20-collusion. Without recursion, the same key assignment cost will be 600 keys per node, 67% more than if recursion is used. Alternatively, we could use a very sparse key assignment (*e.g.*, only one out of 1000 codewords is an actual key-assign ID) to increase overall security with minor (*e.g.*, less than a factor of 10) cost overhead.

VI. CONCLUSION

We propose a symmetric key pre-distribution scheme based on the Maximum-Distance Separable (MDS) codes. The scheme is deterministic, optimal, and can be recursive and used in conjunction with other key distribution schemes. Two trade-offs are investigated against the key assignment costs: one for collusion prevention and one for collusion resilience.

We use Reed-Solomon codes of different lengths as exmaples to calculate actual key assignment costs. In all practical cases, our scheme meets the lower bound $O(c \log_2 N)$ with a multiplicative constant c less than 8. The resilience of our scheme against r-collusion is roughly $1 - R^{d_{\min}}$, where $R = 1 - (1 - 1/q)^{2r}$ and $d_{\min} = n - k + 1$.

VII. ACKNOWLEDGEMENT

We would like to thank Professor Michael Goodrich of Dept. of Computer Sciences at U.C. Irvine for his inputs during the writing of this paper.

REFERENCES

[1] A. S. Aiyer, L. Alvisi, M. G. Gouda, "Key Grids: A protocol family for assigning symmetric keys," Proc. of the 14th IEEE International Conference on Network Protocols (ICNP '06), Nov. 2006, p.178-186.

[2] T. L. Alderson, "Extending MDS codes," Journal of Annals of Combinatorics, vol.9, no.2, p.125-135.

[3] R. Blom, "An optimal class of symmetric key generation systems," EUROCRYPT '84, LNCS 209, Springer-Verlag 1985.

[4] C. Blundo, A. De Santis,A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," CRYPTO '92, LNCS 740, pp. 471-486, 1993.

[5] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," Proc. of the IEEE Symposium on Security and Privacy, p.197-213, 2003.

[6] F. Delgosha, F. Fekri, "Key pre-distribution in wireless sensor networks using multivariate polynomials," IEEE SECON 2005, Sep. 2005.

[7] R. Di Pietro, L. V. Mancini, A. Mei, "Efficient and resilient key discovery based on pseudo-random key pre-deployment," Proc. of the 18th International Parallel and Distributed Processing Symposium, Apr. 2004.

[8] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," ACM Transactions on Information and System Security (TISSEC), May 2005.

[9] M. T. Goodrich, "Leap-Frog packet linking and diverse key distributions for improved integrity in network broadcasts," IEEE Symposium on Security and Privacy, 2005.

[10] S. S. Kulkarni, and B. Bezawada, "A family of collusion resistant protocols for instantiating security," Proc. of the 13th IEEE International Conference on Network Protocols (ICNP '05), 2005, pp. 279-288.

[11] F. J. MacWilliams, N. J. A. Sloane, *The theory of error-correcting codes. II.* North-Holland Mathematical Library, Vol. 16, 1977. pp. i-ix and 370-762. ISBN: 0-444-85010-4

[12] A. Mohaisen, D. H. Nyang, "Hierarchical grid-based pairwise key predistribution scheme for wireless sensor networks," Proc. of the 3rd European workshop on wireless sensor networks, EWSN 2006.

[13] M. Ramkumar, N. Memon, "An efficient random key pre-distribution scheme," IEEE Journal on Selected Areas of Communication, March, 2005.

[14] Silverman, R., "A Metrization for Power-Sets with Applications to Combinatorial Analysis," Canadian Journal of Mathematics, 12 (1960), p.158-176.