Centre driven Controlled Evolution of Wireless Virtual Networks based on Broadcast Tokens

A thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology

by

Atishay Jain , Vignesh Babu
(Roll No. 10010271, 10010261)

Under the guidance of
Dr. Kannan Karthik

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
April 2014

CERTIFICATE

This is to certify that the work contained in this thesis entitled

Centre driven Controlled Evolution of Wireless Virtual Networks based on Broadcast Tokens

is the work of

Atishay Jain , Vignesh Babu
(Roll No. 10010271, 10010261)

for the award of the degree of Bachelor of Technology, carried out in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

_____

Guide

Date: _____

Place: _____

Abstract

In a wireless sensor network, the virtual connectivity between nodes is a function of the keys shared between various nodes. Pre-embedding these key configurations in the nodes would make the network inflexible. On the other hand, permitting subsets of nodes to engage in a common key synthesis phase to create secure distributed connections amongst themselves, would decouple and conceal the information flow from the controlling centre. An intermediate solution is the notion of a centre driven key generation process through broadcast tokens, designed to extract different keys in different nodes based on some prior information stored at the nodes. As more tokens arrive, the virtual connectivity of the nodes are altered and the network evolves. This evolution can be distributed and can be controlled to converge to a certain specific connectivity profile. In this paper we present a framework and an algorithm which controls the simultaneous and distributed key release in different nodes, resulting in the creation of parallel virtual multicast groups. The design of the node shares and the supporting broadcast tokens have been discussed in conjunction with the process of balancing the spans of individual groups with spans of several coexistent multicast groups.

Contents

List of Figures

Chapter 1

Introduction

In collaborative information processing applications, a large computational task is delegated amongst several wireless nodes, by a centre. The main task is split into several sub-tasks and each subtask is assigned to a different group of nodes. Since the computation of each subtask entails a sharing of resources and may require further modularization, the nodes can form virtual groups for exchanging information and also for delegating roles amongst themselves. The information flow across virtual groups, group expansions, group migrations, group mergers, group dissolutions etc. can either be completely distributed or can be triggered by the centre. However when the nodes carry confidential information (related to a large product design) and the overall computational task is also of a sensitive nature, the centre cannot afford to allow the nodes to form their own sub-groups to perform this distributed processing. This would make it difficult for the centre to monitor the flow and also perform intermediate checks on the information fragments exchanged amongst the nodes. Hence, a centre directed group formation and information exchange is recommended.

Secure virtual multicast connections either between subsets of nodes and/or dedicated unicast links between each of the nodes and the centre C, are required to preserve confidentiality of messages exchanged between the nodes within a specific virtual group or between the nodes and the centre. Any secure connection, requires the sharing of an encryption key, which can be pre-distributed by the centre at the time of forming the network and registering new nodes. Alternatively the centre may facilitate the generation of keys in a distributed fashion between several clusters of nodes. The strength of this key establishment phase or key distribution phase is important in determining the overall security of the network. Several key distribution mech-

anisms for Wireless Sensor Networks have been studied in the past.

In pre-distributed shared key mechanisms [3] [4], the centre pre-assigns a set of keys to each node before they are deployed. The key pre- distribution is such that any two neighbouring nodes share a single common key or a group of common keys with a certain probability. In Random pair-wise key schemes [5], a random set of node IDs and corresponding link keys are stored in each node prior to deployment. The nodes broadcast these IDs to advertise the presence of common keys and the connectivity graph is created. An important drawback of these approaches is the degree of uncertainty in connectivity which could lead to disconnected networks. Location based Pair wise key distribution schemes [6] assume that the network topology is known beforehand and the keys stored in each node depend on its neighbours. Hence any two nodes are guaranteed to share a common key. However, these schemes assume that the topology is known prior to deployment, which is rarely the case in practical scenarios. The main problem with these approaches is that physical capture of a node could lead to the loss of many keys and link associations.

Key matrix based schemes described in [7] define a key matrix which stores all the link keys. Each node is given some public and private information derived from the matrix. Neighbouring nodes exchange their public information which is then combined with their respective private information to generate the link key. In Polynomial based key distribution approaches [8], each node stores a partially evaluated symmetric polynomial which is evaluated using the IDs of the neighbouring nodes to establish link keys. Above dynamic key distribution methods, avert problems associated with node-capturing in static pre-distribution schemes. However these approaches are power hungry, computationally intensive and require costly vector multiplications.

Broadcast encryption schemes allow transmission of secrets to a privileged set through a broadcast channel [9]. Broadcast encryption techniques are used in schemes based on combinatorial constructions, one-time revocation schemes based on secret sharing techniques, and tree-based constructions which have been proposed in [10]. Broadcast encryption schemes define a parameter k which specifies that at least k users who do not belong to the privileged set have to collude in order to decrypt the message. Hence they are generally only k-resilient. Moreover they incur high communication cost to broadcast messages, large memory requirement to store

keys in each node and limited group association structures. Further, in all of these schemes, the establishment of group keys for an arbitrary privileged set of size m, would generally require the exchange of at the most m messages resulting in wastage of the available bandwidth. If N_G simultaneous multicast groups must be created, the number of messages required would be anywhere between N_G and $m \times N_G$. Tree based constructions used in Broadcast encryption, have been designed to counter multicast group dynamics and have been optimized for single multicast groups. Simultaneous creation of multiple virtual multicast groups would however require messages proportional to the number of groups.

Another common problem with the above mentioned approaches is that the configuration of the network is static which means that the virtual connections within and between subsets of nodes in the network cannot change with time. However in a collaborative processing application, the topology of the network once deployed may have to be reconfigured to facilitate parallel and distributed computation and also to ensure that intermediate computational results are exchanged amongst different virtual groups. Hence, dynamic re-configurability of the virtual network is paramount. However, when the information handled by the nodes is of a sensitive nature, this re-configuration must be triggered by the centre, to ensure transparency in the information flow.

Another approach, to address this issue of centre driven re-configurability [1], considered an interaction between protected node shares of key blocks and broadcast tokens released by the centre. Here, each wireless node in the network was assigned a protected node-share of an encryption/decryption key set. Upon the release of specially designed tokens which are broadcasted by a centre, the fusion of these shares with the tokens would unlock a set of encryption keys. Common keys unlocked in different nodes can be used to form multicast group associations. The associations can then be easily changed by broadcasting new tokens. The scheme is computationally feasible, fully resilient and physical node capture only results in the loss of the keys which have been unlocked in the captured node. Thus a single broadcast token can result in the formation of several multicast groups, the choice of groups and their evolution can be pre-designed to satisfy a certain function. This pre-design retards the flexibility of the network.

However, it did not specify a design to control the key release and most importantly alter it dynamically to suit a certain network computational goal. A solution which allows the centre to

design tokens that unlock desired keys (controlled key release) in different nodes is imperative to achieve any desired configuration. In this report we propose such a solution based on the protocol described in [1].

The thesis report is organised as follows: In chapter 2, the virtual reconfiguration scheme proposed in [1] is discussed. Chapter 3 provides details of the software implementation of the reconfiguration protocol. In chapter 4, we discuss our initial approach to incorporate controlled key release. In chapter 5, we discuss the framework and the proposed solution which improves up on the solution discussed in chapter 4 and achieves controlled and simultaneous key release. Algorithms which use the proposed solution for efficient Token construction to achieve simultaneous creation of multicast groups is discussed in Chapter 6. Issues pertaining to key leakage and token collusions are discussed in chapter 7. Comparison with tree broadcast encryption and extensions towards forward secrecy are explored in chapter 8. Finally, in chapter 9, some simulation results are presented.

Chapter 2

Virtual Network Reconfiguration

A protocol based on key release mechanisms to reconfigure networks has been discussed in detail in [1]. The entire protocol has been described here for the purpose of clarity which would be useful in understanding subsequent sections. The network is assumed to have a distribution center and a certain number of nodes. The scheme revolves around the idea that a set of encryption keys are locked within protected node-shares in each node of the network. When specially designed tokens are broadcasted by the center, the fusion of these shares with the tokens release a subset of the locked keys. With the arrival of every new token more keys would be released at each node. If the node-shares stored in different nodes are dissimilar, it would imply that different sets of keys could be released by the same token in different nodes. The unlocked keys would then determine the configuration of the network. If a key is common to a set of nodes it would necessarily imply a multicast connection between them in the sense that this key can be used to transmit messages securely within the group formed. With the broadcast of each new token, the configuration of the network changes dynamically, as the associations between different nodes change owing to the release of new keys in each node.

2.1 Description of the model

The model considers a distribution center *C* and n number of nodes say Node 1, Node 2, Node 3, ... , Node n. We assume that all the nodes including the center are in the transmission range of any other node in the network. The center generates a set of v keys say K_1, K_2, K_3,..., K_{v} and a set of shares N_1, N_2, N_3, ..., N_{n} for each node in the network. This means that the i^{th} node is given the node-share N_{i} prior to deployment. The center also generates and

broadcasts specially designed tokens $T_{\mathrm{k}}{}^{'}s$. The node-shares $N_{\mathrm{i}}{}^{'}s$ have complete information required to extract all the v keys using the broadcast tokens $T_{\mathrm{k}}{}^{'}s$. The node-shares $N_{\mathrm{i}}{}^{'}s$ are different for different nodes and every broadcast token T_{k} upon fusion with the node-shares, unlocks different subset of keys in different nodes of the network. This results in the formation of different clusters of nodes each with common shared keys which can be further used for secure multicast communications.

2.2 Generation of node-shares and broadcast tokens

The distribution centre generates these node-shares and tokens using a non-perfect secret sharing scheme called the MIX-SPLIT [11]. It assumes the keys to be uniform length L_{p}-bit random strings. A block X of length ($L_{\mathrm{p}} \times v$) bits is computed as follows. The keys K_1, K_2, K_3, $\ldots,$ K_{v} are first concatenated into a string and interleaved without changing the order of the bits, to form a block X. The partition of a key is defined as the set of bit positions in X that are filled with the bits of that particular key in the same order. Since there are v keys, there would be v disjoint partitions which are designated as $P_1,\ P_2,\ P_3, \ldots,\ P_{\mathrm{v}}$ each of length L_{p}. These partitions are also referred to as hidden partitions as they are unknown to the nodes. Another block Y is defined as the bit complement of X (bit wise not of X).

Macro-mixing of fragments of X and Y is done to produce r preliminary shares say PS_1, PS_2, $\ldots$ PS_{r}. Subsets of fixed size from these preliminary shares in turn form different node share matrices $N_{\mathrm{i}}{}'s$. Each of these preliminary shares can be written in the following form:

$$PS_{\mathrm{i}} = (PS_{\mathrm{i1}}||PS_{\mathrm{i2}}||PS_{\mathrm{i3}}||\ldots PS_{\mathrm{iv}}) \tag{2.1}$$

where the sub-sequence $\mathrm{PS_{ij}}$ is derived according to a pre-defined codebook C defined as

$$C = \begin{pmatrix} c_{11} & c_{12} & \ldots & c_{1v} \\ c_{21} & c_{22} & \ldots & c_{2v} \\ \ldots & \ldots & \ldots & \ldots \\ c_{v1} & c_{v2} & c_{v3} & c_{vv} \end{pmatrix}$$

PS_{ij} is obtained as follows

$$PS_{\mathrm{ij}} = X(P_{\mathrm{j}}) \quad if \ \ c_{\mathrm{ij}} = 1$$

$$PS_{ij} = Y(P_j) \quad if \quad c_{ij} = 0$$

where c_{ij} is an entry of the code book. In general the code book can be partitioned as

$$C = \left(\frac{N}{T} \right)$$

where T is a $t \times v$ matrix whose rows are t code words which are used to construct the broadcast tokens and N is a $(r - - \ t) \times v$ matrix also called as the node-share matrix, whose rows are code words which are used to construct the preliminary shares PS_i. The preliminary node shares and broadcast tokens are built from their respective code words by the above mentioned method. Subsets of these preliminary shares in turn form node-shares which are distributed to all the nodes Node 1, Node 2, Node 3, ... Node n.

The operator $\|$ stands for the concatenation and mixing operation. The values in the bit positions specified by the partition P_j for a particular sub-sequence PS_{ij}, are chosen from X or Y depending on the value of $c_{i,j}$. In other words, the values of the bit positions in PS_i which are specified by P_j will either be equal to the bits of K_j or $K_j{}^c$ depending on the value of c_{ij}. This procedure is then repeated v times for each sub-sequence PS_{ij} to form the preliminary share PS_i. The figures given below illustrate the mixing process for a particular 3 key system where each key is 4 bits long.

	KEY 1				KEY 2				KEY 3			
VALUES	0	1	0	1	1	0	1	1	1	1	0	0
PARTITION	3	5	7	11	1	6	9	12	2	4	8	10

	1	2	3	4	5	6	7	8	9	10	11	12
X	1	1	0	1	1	0	0	0	1	0	1	1
Y	0	0	1	0	0	1	1	1	0	1	0	0

Figure 2.1: An example of partition assignment in a 3 key system

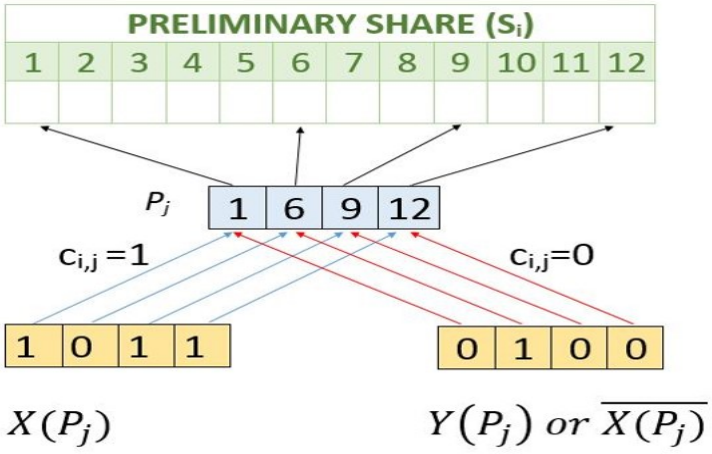

Figure 2.2: Illustration of the key mixing process

2.3 Rules for the design of code books

Since information regarding all the v keys are contained in each of the preliminary shares, it is possible to extract a subset of these keys by stacking selective shares one above the other [2]. The rules for the design of a code book to enforce conditional visibility and invisibility of partitions (for unlocking a subset of keys) are as follows:

Rule 1: Complementary and repetitive columns lead to inseparable partitions i.e, if a set of code words which when stacked over each other form a matrix whose columns are either repetitive or complementary, then the preliminary shares or tokens generated from these code words do not reveal any hidden partitions.

Rule 2: Row-sampling of a complementary pattern is complementary. In other words, if a subset of code words are chosen from a set of code words satisfying the complementary column property, then these subset of code words would also satisfy the complementary column property.

Rule 3: Single shares are always mixed i.e, no partitions are revealed by one share alone. Multiple shares/tokens have to be stacked to reveal partitions.

Rule 4: At least one partition becomes visible if a column is distinct. Hence, if the code-words which are stacked, form a matrix with a unique column, then that corresponding partition and

hence that key, can be unlocked by stacking the actual shares/tokens generated by these code words and searching for that unique column pattern.

The node shares that are distributed to nodes in the network, must not reveal any keys on their own. Hence, the corresponding node share code words should have complementary or repetitive columns (Rule 1). The keys should only be released upon fusion with the broad casted tokens. This means that the stacking of node share code words plus the token code words should form a matrix with one or more unique columns (Rule 4). An additional security requirement that cannot be ignored is that the Tokens themselves should also satisfy Rule 1 to ensure that stacking any subset of Tokens does not reveal hidden partitions to an eavesdropper.

2.4 Revealing hidden partitions

When the stacked code words reveal unique columns, these unique columns can be converted into distinct bit patterns. For instance, if the unique column formed by stacking 3 code words is the binary number $[1\ 0\ 1]^{\mathrm{T}}$, then it represents the bit pattern $[b\ \bar{b}\ b]^{\mathrm{T}}$. In order to reveal the corresponding hidden partition in the actual shares (shares generated by the three code words), this bit pattern is searched on these shares. In other words, the actual shares are traversed column by column and the set of indexes of every column which represents this bit pattern, is the hidden partition [2] [11]. In this example, the hidden partition corresponds to the column indexes in the shares where the following condition is satisfied:

$$Share1(i) \ = \ Share2(i)^{\mathrm{c}} \ = \ Share3(i)$$

All the bit positions i which satisfy this condition form the hidden partition.

2.5 Illustration

Consider a case with a center C and 3 nodes in the network. Let us consider a node-share matrix to be as follows:

$$N = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

It can be observed that the above matrix satisfies Rule 1 i.e, no columns are unique. They are either the same or complementary to each other. Using the above matrix let us construct the node share code word matrices SH_1, SH_2, SH_3 for the 3 nodes as follows:

$$SH_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$SH_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$SH_3 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

It can be noted here that these matrices are obtained by considering code word subsets of size 2. It is again imperative that the broadcast tokens T_k's also satisfy Rule 1 as they are exposed to traitors not belonging to the network. Our aim is to release some of the keys upon the fusion of tokens with the node shares. Hence we consider a token generator matrix with repetitive columns (to satisfy Rule 1). All the token code words obey the following format.

$$T = \begin{pmatrix} z_1 & z_2 & z_3 & z_2 & z_3 & z_1 \end{pmatrix}$$

The total number of arrangements of the bit-positions is $2^3 = 8$, the total number of unique tokens satisfying Rule 1 is 8. Now this number also includes those token code words which are bit complimentary with respect to other tokens. These tokens will not change the stack relation and are redundant. Hence the number of useful tokens is $2^3/2 = 4$. Accordingly, the token generator matrix can be represented as follows:

$$T = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where each row represents a broadcast token code word. The actual broadcast tokens are generated from these code words. When a Token for example T_1 is broadcasted , at Node 1 it

will fuse with the node-share corresponding to Node 1 which is SH_1 and will give rise to the following three distinct stack equations:

$$\begin{pmatrix} b \\ b \\ b \end{pmatrix}, \begin{pmatrix} b \\ b \\ \bar{b} \end{pmatrix}, \begin{pmatrix} b \\ \bar{b} \\ b \end{pmatrix}$$

$$T_1 + SH_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Out of these three equations only one is unique which is

$$\begin{pmatrix} b \\ \bar{b} \\ b \end{pmatrix}$$

This implies that the partition P_4 will be made visible. To obtain key 4 corresponding to this partition we traverse through the stack of actual shares/tokens (generated using the MIX SPLIT method by substituting the code-word bits by the actual bits of X or Y) and check for the pattern given above [11]. The indexes which satisfy this pattern will give us the desired bit locations. The values at these bit locations when concatenated will give us either key K_4 or its compliment. Hence key 4 is unlocked. Similarly, this method is followed in all the other nodes as well. The unlocked partitions in Node 2 and Node 3 are 6 and 4 respectively. The set of unlocked keys at each node increases with the arrival of new tokens. The table below shows the sequential unlocking process with the arrival of each token.

Table 2.1: Cumulative list of keys unlocked at each node

Token no	**Node 1**	**Node 2**	**Node 3**
1	K_4	K_6	K_4
2	K_3,K_4	K_3,K_6	K_1,K_3,K_4
3	K_2,K_3,K_4,K_5	K_3,K_4,K_6	K_1,K_3,K_4
4	K_2,K_3,K_4,K_5	K_3,K_4,K_5,K_6	K_1,K_2,K_3,K_4

Chapter 3

Implementation

3.1 Design environment and testing

We implemented the protocol on the contiki-2.6 operating system. It has been simulated on the IEEE 802.15.4 based TMote Sky platform. The Cooja simulator that comes in built with the Contiki OS, was used for simulating the wireless sensor network (WSN). The programming was done in C language.

3.2 Implementation specifics

- *Inbuilt code book*: The code book is stored in the distribution centre prior to the deployment of the network.
- *Initial Node share distribution*: In our implementation, after the network simulation is started, the centre computes the Node shares and unicasts them to the respective nodes. We assume that network is deployed only after the initial node shares have been distributed by the centre. Hence, the eaves dropper cannot obtain these node shares.
- *Token Broadcast*: The centre broadcasts the Tokens periodically. Hence, the configuration of the network will change at regular intervals.
- *Encryption/Decryption algorithm*: We implemented and tested the Tiny Encryption Algorithm (TEA) along with this protocol. This symmetric key encryption/decryption algorithm is optimum for small wireless device security. It can be used for actual encryption/decryption of messages that are exchanged between nodes once the reconfiguration

protocol unlocks some common keys. Provisions have been made for the messages to be encrypted and sent in numbered fragments. Acknowledgements are sent back to the sender upon successful reception. Duplicate packets are also removed automatically.

3.3 Simulations

We simulated the protocol for a 6 key 4 node network. One of the nodes in the network was programmed to behave as the distribution centre. All the nodes were within the transmission range of each other. We designed a code book and verified the pattern of keys released with each new broadcast token. The protocol was observed to function properly and changes in the configuration changes with each Token were observed to be correct.
After a thorough investigation, we were able to identify the main strengths and weaknesses of the described scheme.

3.4 Protocol analysis

Strengths:
The main strengths of the protocol are:

- Enables virtual re-configurability of the network by using a center driven broadcast system. For example, it can be used in a WSN to re-distribute load adaptively within the network.
- For a network with a large number of keys, the damage caused by capturing certain nodes can be reduced since only a small subset of keys will be compromised.
- Secure additions of nodes to a multicast group without leakage of information.
- Secure unicast connections with the center are also possible.

Drawbacks:

- *Configuration Control* - There is no control over the keys which are unlocked by the tokens in each node and hence we have no control over the configuration change.

- *Possible Configurations* - The number of tokens that can be broadcasted, are limited by the number of keys. For example only 4 tokens are possible for 6 keys. Since we do not have control over the release of keys and tokens are easily exhausted only a few and not all of the possible configurations can be achieved.

- *Configuration Invariability* - The configuration of the network cannot be changed after the exhaustion of all the tokens or in other words it freezes after some amount of time.

- *Configuration Re-traceability* - We cannot revert back to a previous configuration after the arrival of the token.

To overcome some of these drawbacks we initially proposed a solution. The solution uses a new codebook design along with rules to generate node-shares and broadcast tokens from this codebook. It has been described in detail in the following chapter along with a mathematical proof and examples.

Chapter 4

Selective Unlocking Mechanism

We understood that one possible solution to the Token exhaustion problem would be to design tokens and node shares such that each token primarily unlocks only one key in one of the nodes in the network. We worked towards developing such a solution which would allow the distribution centre to control which key gets unlocked in which node in the network.

4.1 Node share matrix design

In order to realize the above mentioned solution, we propose to design a Node share matrix with the following distance property.

Definition 1. ***Distance property****: Let v be the number of keys. Then the hamming distance H between any two pairs of the generated node share code words should lie between*$[4,\ v\ -\ 4]$. *In other words, let N_i and N_j be two node share code words. Then,*

$$4 \leq\ H(N_i, N_j)\ \leq v\ -\ 4$$

4.1.1 Generation of Node share matrix with code words satisfying the Distance property

We briefly present a simple method to generate node share code words satisfying the Distance property. For all future purposes we use the following notations:

- *v* represents the number of keys

- *n* represents the maximum number of nodes that can be supported by a '*v*' key system
- *N* represents an $n \times v$ Node share matrix
- N_i represents the i^{th} node share code word or the i^{th} row of the Node share matrix N
- *S* is an $n \times (v/2)$ matrix such that *N*= $[S \;\; S^c]$
- S_i represents the i^{th} row in *S*

We consider a $n \times (v/2)$ matrix *S*.

1. To construct *S* we start with a partial code word S_1 with equal number of zeros and ones i.e, S_1 has (*v*/4) zeros and (*v*/4) ones. S_1 is the first row of *S*.
2. To obtain the other rows we permute the bits of S_1 and consider only the permutations which are not complementary to each other i.e, no two rows S_i and S_j should be complementary.
3. To obtain the Node share matrix, we concatenate *S* and its complement.

Note:

- It can be observed that the maximum number of such non complementary permutations equals n. Hence,
$$n \;=\; (v/2)!/2(v/4)!(v/4)!$$
- The code words generated by this method are *n* out of *n* collusion secure.

4.1.2 Proof that these code words satisfy the Distance property

To prove the Distance property for these code words, we consider two partial code words S_i and S_j. Without loss of generality, let us assume that S_j is obtained by a permutation on the bits of S_i. Since S_i and S_j have equal number of 1s and 0s, the least hamming distance that could be obtained by a permutation is 2. Hence,

$$H(S_i, S_j) \;\geq 2$$

Similarly, since S_i and S_j cannot be complements of each other, using the same equal number of 0s and 1s argument, we can prove that the maximum Hamming distance between them can be at most $(v/2) - 2$. Hence,

$$H(S_i, S_j) \leq (v/2) - 2$$

Since N_i and N_j are just concatenations of S_i, S_i^c, S_j and S_j^c respectively, we have

$$4 \leq H(N_i, N_j) \leq v - 4$$

Hence, the satisfaction of distance property has been proved for these code words.

4.1.3 Illustration: Node share matrix generation for a 12 key system

Since v=12, we get n=10 from the equation derived above. Let $S_1 = [0\ 0\ 0\ 1\ 1\ 1]$, then S is given by:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

The Node share matrix is then obtained as $N = [S\ \ S^c]$

4.1.4 Node share distribution

In this scheme, each node is given one node share. For example, the node share of node i is generated using the i^{th} row of the Node share matrix i.e, N_i becomes the node share code word for node i.

4.2 Token design

Let us say that the centre wants to unlock key '*j*' in node '*i*' without unlocking any other key in any other node. Let the corresponding token be denoted by T_{ij}. Then T_{ji} is given by:

$$T_{ji} \ = \ N_i \oplus I_j$$

where I_j is the j^{th} row of a $v \times v$ identity matrix *I*. In other words, the j^{th} bit of N_i is flipped to get T_{ji}.

4.2.1 Proof of desired token behaviour

In this proof, we use the rules for conditional visibility and invisibility of partitions which have been described earlier. Since only one bit of N_i gets flipped to get T_{ji}, we have $H(N_i,T_{ji}) \ = \ 1$. But we already know that,

$$4 \leq \ H(N_i, N_k) \ \leq v \ - \ 4 \ \forall \, k \ \neq \ i$$

Since one bit flip to N_i can at most reduce the Hamming distance between N_i and N_k by 1 or increase it by at most 1, we get

$$3 \leq \ H(T_{ji}, N_k) \ \leq v \ - \ 3 \ \forall \, k \ \neq \ i$$

Since the Hamming distance between T_{ji} and N_k is at least 3, when T_{ji} is stacked over N_k there would be at least three columns satisfying the equation $[\ \bar{b} \ \ b\ ]^T$.

Similarly since the Hamming distance between T_{ji} and N_k is at most *v* - 3, when T_{ji} is stacked over N_k there would be at least three columns satisfying the equation $[\ b \ \ b\ ]^T$

Hence there would be no unique equation that is revealed when T_{ji} is stacked over $N_k \ \forall k \neq i$ So no keys will be unlocked in any other node. However, when T_{ji} is stacked over N_i, the column which is given by the equation $[\ \bar{b} \ \ b\ ]^T$ becomes unique. Hence Key '*j*' would be unlocked in node '*i*'.

4.2.2 Illustration

Suppose, we need to unlock key 5 in node 6 in a 12 key system. We use the same Node share matrix presented before in this example. T_{56} equals:

$$T_{56} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

By stacking T_{56} over each N_i, it can be easily verified that Key 5 will be unlocked only in node 6 and no keys will be unlocked in any other node.

4.2.3 Advantages of the proposed code book design

- Many Tokens can be generated and it is possible to achieve any network configuration. In fact upto $n*v$ different Tokens can be generated and n^v configurations can be achieved.
- Node shares and Tokens are easy to generate.
- Node shares are n out of collusion secure i.e, even if all the node shares are stacked over each other, no partition would be revealed.
- For larger values of v, a much larger network can be supported.

Chapter 5

Proposed scheme

The method of release of keys described in [1] ensures dynamic reconfiguration. However, it poses the challenge of control over the node associations. In the previous chapter, we proposed a solution which incorporates control in forming these node associations. However, the code book design had other issues. The tokens generated are only 2 collusion secure which means that by combining more than two tokens, the eaves dropper could potentially learn the hidden partitions. Another major drawback of the previouly proposed solution, is the bandwidth limitations of the tokens. They do not support simultaneous release of multiple keys in different nodes and hence the tokens waste a lot of bandwidth.

One of the common points that one could notice in the two schemes is that the partitions allotted to the keys are fixed and do not vary from one node share to another. The token design is constrained by the requirement to maintain and collusion security among tokens. The difficulty in incorporating controlled key release over this frame work stems primarily from static partition allotment which in turn forces any two tokens to be related. Our next proposal differs from these two schemes in this aspect. We vary the partitions to which different keys are allotted and design tokens which are unrelated to each other. We show that our solution achieves controlled and simultaneous key release without compromising the token security.

5.1 Description:

The setup of the network is similar to the one described in Section 2. The network comprises of a center D and nodes Node 1, Node 2, ..., Node n all within the transmission range of each

other. Let K_1, K_2, K_3,..., K_v be v keys each of length L_p, generated by the center. Let n be the maximum number of nodes that can be supported by a 'v'-key system and M represent the number of preliminary shares given to each node, i.e, for any node i , the set of node share code words would be $N_i = N_{i1}, N_{i2}, \ldots, N_{iM}$

Each key K_i in the set of v keys is divided into two halves K_{i1} and K_{i2}. Each half is locked in a hidden partition. Hence the size of each codeword is $(1 \times 2v)$. The first half of the keys, $K_{i1}\ \forall i = 1, 2, \ldots, v$ are encoded using the first v bits of the code word while the second half, $K_{i2}\ \forall i = 1, 2, \ldots, v$ are encoded using the next v bits of the code word.

5.2 Formulation

Prior to the code word construction, each Key K_i is assigned a fixed number such that no two keys are assigned the same number or its complement in binary. It is implied that since each key is assigned a fixed number, both the halves of the key are implicitly assigned the same number. For example, in a 4 key system, the following assignment could be made:

Table 5.1: Key Table

Key	**Number Assigned**
K_1	1
K_2	2
K_3	3
K_4	7

M is calculated as the number of bits required to represent the largest number in the key table. From the table shown above, it can be deduced that the value of M is 3 in this case.

5.3 Code word representation

Here we describe a convenient way to represent Node share code words. Consider a set of 3 preliminary share code words (which together are used to generate the node share for a particular node i) representing a 4 Key system (8 partitions numbered from 1 to 8). It must be noted that these set of code words have repetitive columns (Rule 1). Hence the node shares generated

from these code words do not reveal any keys.

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{matrix}$$

If these code words are read column wise, it is easy to see that they represent the following sequence of numbers:

$$N_{\mathrm{i}} = 7\ 2\ 3\ 1\ 2\ 3\ 1\ 7$$

If it is assumed that the above stated Key table is used in the mapping, then this means that, in this set of code words the following mapping has taken place :

Table 5.2: Partition Assignment

Key (Half)	**Assigned Number**	**Partition No.**
K_{11}	1	4
K_{12}	1	7
K_{21}	2	2
K_{22}	2	5
K_{31}	3	3
K_{32}	3	6
K_{41}	7	1
K_{42}	7	8

A particular Key K_{j} in the node i, can be represented by the positions of its two halves. In the above example:

$$K_1{}^{\mathrm{i}} = (4,7),\ K_2{}^{\mathrm{i}} = (2,7),\ K_3{}^{\mathrm{i}} = (3,6),\ K_4{}^{\mathrm{i}} = (1,8)$$

5.4 Unique mapping property

If a particular key K_{i} in node j is represented by the coordinates (x, y) i.e,

$$K_{\mathrm{i}}{}^{\mathrm{j}} \ = \ (x_{\mathrm{i}}{}^{\mathrm{j}}, y_{\mathrm{i}}{}^{\mathrm{j}})$$

Then no other node has key K_i in the positions (x_i^j, y_i^j) .This must be true for all $K_i{}'s$. Hence,

$$\{K_i^p \neq (x_i^j, y_i^j)\ \forall p \neq j\}\ and\ \forall\ i$$

5.5 Node share code word generation

In order to generate node shares N_i satisfying the Unique mapping property, we define two matrices C_1 and C_2 which are both $(v \times v)$ matrices whose rows are used to construct the node share code words. Both matrices C_1 and C_2 are generated by the following algorithm. Let $C_i[j]$ represent the j^{th} row of the matrix C_i.

Algorithm 1 Construction of C_1 and C_2

Start with a $(1 \times v)$ random permutation (R) of the numbers in the key table

for $j \leftarrow 1\ to\ v$ **do**

$C_i[j] \leftarrow R$

Circular right shift R by 1

end for

$Illustration$:

Let the number of keys be 4. Refer to Table I for the mapping of keys. Let C_1 represent a matrix containing first half of node share code words formed using a random seed R_1 say:

$$R_1 = \begin{bmatrix} 7 & 1 & 2 & 3 \end{bmatrix}$$

C_1 is then given by:

$$C_1 = \begin{bmatrix} 7 & 1 & 2 & 3 \\ 3 & 7 & 1 & 2 \\ 2 & 3 & 7 & 1 \\ 1 & 2 & 3 & 7 \end{bmatrix}$$

Similarly, the other matrix C_2 can be obtained with another random seed say

$$R_2 = \begin{bmatrix} 3 & 2 & 1 & 7 \end{bmatrix}$$

C_2 is then given by:

$$C_2 = \begin{bmatrix} 3 & 2 & 1 & 7 \\ 7 & 3 & 2 & 1 \\ 1 & 7 & 3 & 2 \\ 2 & 1 & 7 & 3 \end{bmatrix}$$

After the obtaining both matrices C_1 and C_2, the node share code word for any node with ID i is constructed as follows,

$$Let\ q = Quotient(i/v);\ \ r = (i\ mod\ v),\ then,$$

$$N_{\mathrm{i}} = C_1(q) \parallel C_2(r) \tag{5.1}$$

This procedure could generate a maximum of v^2 node share code words. Hence, at most v^2 nodes can be supported by a v key system. It can be easily seen that the node shares generated by this procedure are n out of n collusion secure. Both random seeds R_1 and R_2 contain exactly one instance of every number in the key table. Thus every node share code word contains exactly two instances of every number in the key table. From Rule 1, one can infer that the code words have repetitive columns and they do not reveal any hidden partitions.

Example:

Let us find the node share code word for Node 7. Since Node ID = 7 and $v = 4$, we have $q = 1, r = 3$. Then,

$$N_7 = \begin{bmatrix} 3 & 7 & 1 & 2 & 2 & 7 & 1 & 3 \end{bmatrix}$$

Note: Appendix I proves that code words generated using this procedure satisfy the Unique Mapping property.

5.6 Token design

The unlocking of desired partitions in one or more nodes is done with the help of broadcast tokens. The tokens are fabricated in a way so as to unlock the corresponding partitions without the release of undesired partitions in nodes other than the target nodes.

Illustration:

Referring to the example stated earlier, for a 4-key system let the node shares in two nodes i and j be given by:

$$N_{\mathrm{i}} = \begin{bmatrix} 7 & 2 & 3 & 1 & 2 & 3 & 1 & 7 \end{bmatrix}$$

$$N_{\mathrm{j}} = \begin{bmatrix} 7 & 2 & 3 & 1 & 3 & 2 & 7 & 1 \end{bmatrix}$$

Suppose we want to unlock the partitions corresponding to the number 7 ,i.e, we would like to release key K_4 in nodes i and j. Then they are represented by the following points:

$$K_4{}^{\mathrm{i}} = (1, 8)$$

$$K_4{}^{\mathrm{j}} = (1, 7)$$

This can be achieved effectively by the design of proper token. The token T is designed as:

$$T = \begin{bmatrix} K_{41} & K_{\mathrm{R}} & K_{\mathrm{R}} & K_{\mathrm{R}} & K_{\mathrm{R}} & K_{\mathrm{R}} & K_{42}{}^{,} & K_{42}{}^{,} \end{bmatrix}$$

$$P = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \end{bmatrix}$$

where K_{41} is the first half of K_4 and $K_{42}{}^{\mathrm{c}}$ is the bit-compliment of the second half of K_4, and K_{R} is a random L_{p} bit binary number. P represents the corresponding partitions. In this example, K_{41} is filled in partition 1 i.e P_1. $K_{42}{}^{\mathrm{c}}$ is filled in partitions 7 and 8. The other partitions are filled with random L_{p} bit numbers K_{R}

When T fuses with N_{i} and N_{j} , stack relation equations corresponding to K_{41} and K_{42} become unique. In this example, the equation for K_{41} will be $[\, b\ b\ b\ b\,]^{\mathrm{T}}$ (which corresponds to the number $[1\ 1\ 1\ 1\,]^{\mathrm{T}}$) whereas the stack equation for K_{42} is $[\, \bar{b}\ b\ b\ b\,]^{\mathrm{T}}$ (it corresponds to the number $[0\ 1\ 1\ 1]^{\mathrm{T}}$) in both nodes i and j.

In other words, the stack equation of P_1 in both nodes would be:

$$\begin{bmatrix} b & b & b & b \end{bmatrix}^{\mathrm{T}}$$

and the stack equation of P_7 in node j and the stack equation of P_8 in node i would be:

$$\begin{bmatrix} \bar{b} & b & b & b \end{bmatrix}^{\mathrm{T}}$$

In all the other partitions (other than P_1 and P_7 in node j and P_1 and P_8 in node i), no key would be released. This is because the token holds a random number in these partitions which is not related to any of the keys residing in them. Hence the stack equations would be invalid for these partitions.

Further, since the unique mapping property is satisfied, no other node will correspond to these points $(1, 8)$ and $(1, 7)$ in the Key K_4 space . In all the other nodes, the stack equations would be invalid for at least one of the halves of the key K_4. Hence at least one of the halves cannot be unlocked in any other node. (In fact both halves cannot be unlocked in every other node. It will be subsequently proved). This results in the simultaneous controlled release of Key K_4 only in nodes i and j.

$$K_4 \;=\; K_{41} \;\|\; K_{42}$$

5.7 General procedure

- Suppose K_{i} is to be unlocked in nodes p and q. Let $K_{\mathrm{i}}{}^{\mathrm{p}} = (x_{\mathrm{i}}{}^{\mathrm{p}}, y_{\mathrm{i}}{}^{\mathrm{p}})\ and\ K_{\mathrm{i}}{}^{\mathrm{q}} = (x_{\mathrm{i}}{}^{\mathrm{q}}, y_{\mathrm{i}}{}^{\mathrm{q}})$

- Then if at least one of the coordinates are equal i.e, $(x_{\mathrm{i}}{}^{\mathrm{p}} = x_{\mathrm{i}}{}^{\mathrm{q}}$ or $y_{\mathrm{i}}{}^{\mathrm{p}} = y_{\mathrm{i}}{}^{\mathrm{q}}$, the token is constructed as follows :

  K_{i1} is filled in partition numbers $x_{\mathrm{i}}{}^{\mathrm{p}}$ and $x_{\mathrm{i}}{}^{\mathrm{q}}$, $K_{\mathrm{i2}}{}^{\mathrm{c}}$ is filled in partition numbers $y_{\mathrm{i}}{}^{\mathrm{p}}$ and $y_{\mathrm{i}}{}^{\mathrm{q}}$.

- This idea can easily be extended to more than two nodes as well, provided at least one of the coordinates is the same for all nodes in the privileged set. In other words, suppose if the key K_{i} is to be released in the set of nodes $S = (x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_{\mathrm{N}}, y_{\mathrm{N}})$ such that $x_1 = x_2 = \ldots x_{\mathrm{N}} = x$ and y_{i} are all distinct, a single token T can be constructed as follows:

  K_{i1} is filled in partition number x and $K_{\mathrm{i2}}{}^{\mathrm{c}}$ is filled in partition numbers $y_{\mathrm{j}}\ \forall\ j\ =$ 1, 2, ..., N. A similar approach can be used for the alternative scenario where all the y coordinates are the same and all the x coordinates are distinct for the members of the privileged set.

  Note: Appendix II proves that the above stated procedure indeed achieves controlled and simultaneous key release.

- It must be noted in the above stated general procedure that if both $(x_{\mathrm{i}}{}^{\mathrm{p}} \neq x_{\mathrm{i}}{}^{\mathrm{q}}$ and $y_{\mathrm{i}}{}^{\mathrm{p}} \neq y_{\mathrm{i}}{}^{\mathrm{q}}$, then the key will be unlocked in the desired nodes p and q as well as two other nodes

corresponding to the points $(x_i{}^p, y_i{}^q)$ and $(x_i{}^q, y_i{}^p)$. Hence two separate messages must be sent to avoid this.

- In other words, a single message can unlock a key in a set of nodes or points (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) if and only if one of the coordinates (x_i or y_i) are equal for all the points i.e the points lie on a line parallel to the x or y axis in a 2-D Grid. If this is not the case, then the set of points can be broken into disjoint subsets each satisfying the above stated requirement, and a token can be constructed for each subset to unlock the key in all the nodes of that subset. The number of tokens required will then be equal to the number of subsets. In the worst case, the number of subsets would be equal to the number of points N which would in turn translate to N tokens.

- Further, if one of the coordinate say **x** is fixed, then **y** can take at most **v** values. This means that a single token can unlock one key in a maximum of v nodes simultaneously.

- It must also be noted that two different keys K_i and K_j can also be unlocked simultaneously in two different nodes p and q respectively, provided both the coordinates are such that $x_i{}^p \neq x_j{}^q$ and $y_i{}^p \neq y_j{}^q$. This can be easily extended to more than two nodes as well.

- Another observation that can be made is that more than one key can be unlocked simultaneously in a particular node with a single token. In fact up to all the **v** keys can be unlocked in a particular node with a single token.

Chapter 6

Token construction

One of the goals of the reconfiguration protocol is to minimize the number of tokens and achieve the desired configuration. In the previous section, we showed that a single token can unlock the same key or different keys simultaneously in many nodes. This property of the design can be used to reduce the number of tokens. This section describes a sub optimal greedy algorithm which makes maximum utilization of each token. The algorithm uses two functions which are analysed first.

6.1 Choosing points from a Key grid

Any desired configuration can be achieved by unlocking different subsets of keys in different nodes. Since each key is represented by a point in that particular key space/grid depending on the node at which it is to be unlocked, the whole configuration can be considered as a set of v key grids G_1, G_2 ..., G_{v} with each key grid G_{i} containing a set of points where each point represents a node at which the Key $\mathrm{K_i}$ has to be unlocked. Let the desired configuration be represented by G, then,

$$G = \{G_1,\ G_2,\ \ldots\ , G_{\mathrm{v}}\}$$

where, $G_{\mathrm{i}} = \{p_1,\ p_2,\ \ldots\ ,\ p_{\mathrm{n_i}}\}\ and\ |G_{\mathrm{i}}| = n_{\mathrm{i}}$

Let $p_{\mathrm{i}}[x]$ and $p_{\mathrm{i}}[y]$ represent the x and y coordinates of any point p_{i}. Let $S_{\mathrm{X}}[x_{\mathrm{k}}]$ be the set of points from a given key grid with their x coordinate equal to x_{k}. Similarly, let $S_{\mathrm{Y}}[y_{\mathrm{k}}]$ be the set of points from a given key grid with their y coordinate equal to y_{k}.

Definition 2. *A valid subset of points in a key grid G_i is any subset of points belonging to G_i with all points in the subset having the same x coordinate or the same y coordinate. In other words, all points in a valid subset lie on lines parallel to either the x axis or the y axis.*

Algorithm 2 $find\_largest\_valid\_subset(G_{\mathrm{i}})$

$S_{\mathrm{X}} \leftarrow \emptyset$

$S_{\mathrm{Y}} \leftarrow \emptyset$

for $j \leftarrow 1$ to $|G_{\mathrm{i}}|$ **do**

$P_{\mathrm{j}} = G_{\mathrm{i}}(j)$

$S_{\mathrm{X}}(P_{\mathrm{j}}[x]) \ = \ S_{\mathrm{X}}(P_{\mathrm{j}}[x]) \bigcup P_{\mathrm{j}}$

$S_{\mathrm{Y}}(P_{\mathrm{j}}[y]) \ = \ S_{\mathrm{Y}}(P_{\mathrm{j}}[y]) \bigcup P_{\mathrm{j}}$

end for

$S_{\mathrm{Xbest}} \ = \ S_{\mathrm{X}}(\lambda) \ \ s.t \ \ \lambda \ = \ \underset{x}{\operatorname{argmin}} |S_{\mathrm{X}}(x)|$

$S_{\mathrm{Ybest}} \ = \ S_{\mathrm{Y}}(\gamma) \ \ s.t \ \ \gamma \ = \ \underset{y}{\operatorname{argmin}} |S_{\mathrm{Y}}(y)|$

if $|S_{\mathrm{Xbest}}| > |S_{\mathrm{Ybest}}|$ **then**

$return \ \ S_{\mathrm{Xbest}}$

else

$return \ \ S_{\mathrm{Ybest}}$

end if

Algorithm 2 finds the largest valid subset in any key grid G_{i}. It has a running time of $O(|G_{\mathrm{i}}|)$ which is $O(|G|/v)$ on an average.

6.2 Building a token

The following algorithm (Algorithm 3) fills the partitions in a token with key halves. It implements the design principles described in the previous section. Let the T be the token and let $T[i]$ represent the i^{th} partition in T i.e, P_{i}. The algorithm takes in an argument a set S which is a valid subset in the key space of a particular key say K_{i}.

The running time of the algorithm is $O(|S|)$ which is reasonably small.

Algorithm 3 $Build\_token(T, S, K_{\mathrm{i}})$

for $i \leftarrow 1$ to $|S|$ **do**

$p_{\mathrm{i}} \leftarrow S[i]$

$T(p_{\mathrm{i}}[x]) \leftarrow K_{\mathrm{i1}}$

$T(p_{\mathrm{i}}[y]) \leftarrow K_{\mathrm{i2}}{}^{\mathrm{c}}$

end for

6.3 Token construction

The algorithm described in this subsection (Algorithm 4) uses the Algorithms 2 and 3 to construct tokens and eventually achieve the desired configuration G. Before presenting the pseudo code, two important definitions which are used by the algorithm are stated below.

Definition 3. *Consider a key grid containing a set of points. A grid span β_{R,G_i} of any set of points R, is defined as*

$$\beta_{R,G_i} \ = \ \{p \in G_i \ s.t \ \exists c \in R \ \ p[x] = c[x] \ \ OR \ \ p[y] = c[y]\}$$

In general, for any two sets A and B,

$$\beta_{B,A} \ = \ \{p \in A \ s.t \ \exists c \in B \ \ p[x] = c[x] \ \ OR \ \ p[y] = c[y]\}$$

Definition 4. *The operation $*$ is defined on two sets A and B as follows,*

$$A * B = A \ - \ \beta_{B,A}$$

In other words, the operation $*$ finds the set of points in A which do not lie in the grid span of B.

The algorithm progresses from one key grid to the next and at each step it finds the largest valid subset among points in the key grid which do not belong to the grid span of any of the previously computed largest valid subsets of the other key grids. The algorithm assumes that the desired configuration G is available. At each iteration, a set R stores points (pairs of partitions) which are to be filled by keys in order to construct the Token. The set R is updated from one key grid G_{i} to the next by adding all points belonging to the largest valid subset of $G_{\mathrm{i+1}} * R$ i.e, by combining the largest valid subset computed from points in $G_{\mathrm{i+1}}$ which do not lie in the grid span of any of the previously computed largest valid subsets.

Algorithm 4 Token_construction

$j \leftarrow 1$

while $G \neq \emptyset$ **do**

$T_{\mathrm{j}} \leftarrow Random\_Init(|T|)$

$R \leftarrow \emptyset$

for $i \leftarrow 1$ to v **do**

if $No\ further\ points\ can\ be\ added\ to\ T$ **then**

$Transmit(T_{\mathrm{j}})$

Jump to START

else

if $G_{\mathrm{i}} \neq \emptyset$ **then**

$R_{\mathrm{i}} \leftarrow find\_largest\_valid\_subset(G_{\mathrm{i}} * R)$

$G_{\mathrm{i}} \leftarrow G_{\mathrm{i}} - R_{\mathrm{i}}$

$Build\_token(T_{\mathrm{j}}, R_{\mathrm{i}}, K_{\mathrm{i}})$

$R \leftarrow R \bigcup R_{\mathrm{i}}$

end if

end if

$i \leftarrow i + 1$

end for

$Transmit(T_{\mathrm{j}})$

START : $j \leftarrow j + 1$

end while

The algorithm has a run time complexity of $O(|G|^2)$. Our model assumes a powerful distribution centre which constructs tokens. Hence the burden of computation falls on the distribution centre and computational costs at the nodes are minimal.

Chapter 7

Security

7.1 Collusion resistance

Any set of nodes which do not belong to the privileged set cannot collaborate to reconstruct key halves. Let us consider a scenario where a token releases key K_{i} in a privileged set. Some bits of the key halves Ki1 and Ki2 would be released in the other nodes. Can they collude and reconstruct the key halves ?

The answer is that it would be difficult to do so in polynomial time. Each token is randomly initialised and hence the random numbers filled will be differ from one partition to another within the same token. So the possible "bit values" of K_{i1} and K_{i2} that could be revealed (in a node which does not belong to the privileged set of K_{i}) would also vary from one node to another. The relative position of these bits would also vary because the key halves would occupy different partitions in different nodes. Since both the bit values as well as their positions differ from one node to another, it would be difficult for the nodes to collude because they would not know which bit values would have to be combined to get the key halves. Due to variation of the bit positions, they should not be able to guess the order of bit combination as well. Further there is an added complexity because only some bits of K_{i1} and K_{i2} will be released and they would be revealed along with a union of some random bits. These random bits would also vary from one node to another because of the random token initialisation. So two colluding nodes cannot find out which bits in the released Union (some bits of K_{i1} or K_{i2} + some random bits) belong to the key halves because both (bits of K_{i1} or K_{i2} and random bits) would vary from one node to another.

7.2 Token security

Since the tokens consist of two partitions for each key-node pair which are randomly distributed, the formation of any unique columns (if any) by stacking more than one token together will not lead to the release of a key. When a key is released in a particular node i.e, when the partitions corresponding to the point represented by the node are filled with the key halves, no other subsequent token would have the same key halves in the same two partitions. Hence unique columns cannot be formed by stacking because the partitions that a particular key occupies keeps varying from one token to another. The tokens formed are unrelated to each other and thus they are collusion-secure. The scheme can be made less sensitive to physical node capture by introducing some additional steps. Since both halves of a key are unlocked separately to obtain the key, we can ensure that the node deletes all bits of the node share that lie in the union of these two partitions. The node then stores the union of the two partitions. Before processing tokens, the node simply removes all bits in the token that lie in the union of the two partitions. The other partitions remain intact and the stack relations do not change. The attacker who captures the node can only recover the unlocked key and the union of its corresponding two partitions. If he is able to identify the hidden partitions, then he could use the previously transmitted tokens to obtain information about other keys which could have resided in the compromised partitions. By storing only the union of the two hidden partitions, we ensure that the attacker cannot obtain the two halves separately in polynomial time or gain information about other keys from the past or future token transmissions.

Chapter 8

Discussions

8.1 Properties of Tokens

Each token can carry at most v keys which are not necessarily unique. For each token, we define the total instance of a particular Key K_{i} as the number of nodes in which the token releases the key K_{i}. It is represented by the symbol $I_{\mathrm{K_i}}$. If the key K_{i} is not released by the token in any node, $I_{\mathrm{K_i}}$ is set to zero. The tokens can be classified based on their bandwidth utilization, into two types : Efficient and Inefficient. The classification depends on the total number of instances of all keys released by the token i.e, the classification is done using the quantity $|K| = \sum_{\mathrm{i}} I_{\mathrm{K_i}}$

Efficient Tokens: Tokens which release atleast v instances of one or more keys are termed as Efficient ($|K| \geq v$). A single token can simultaneously release at most $2v-2$ instances in total of two or more keys i.e $|K|_{\mathrm{max}} = 2v-2$. For instance, simultaneous release of $2v-2$ instances of two keys (say K_{i} and K_{j}) is possible if two non overlapping valid subsets (of K_{i} and K_{j} respectively) each of size $v-1$ ($I_{\mathrm{K_i}} = v-1$ and $I_{\mathrm{K_j}} = v-1$) are grouped together into one token. As an example, in a 3 key system, suppose the privileged set of K_{i} is S_{i} = $\{(1,4),(1,5)\}$ and the privileged set of K_{j} is S_{j} = $\{(2,6),(3,6)\}$, then the simultaneous release of K_{i} and K_{j} in S_{i} and S_{j} respectively, can be achieved by a single token. This is because the privileged sets are non overlapping i.e the partitions occupied by these privileged sets do not intersect. In this example, although the total size of the transmitted token is only three key lengths (since $v = 3$), it is able to unlock $|K| = 4$ instances of keys (2 instances of each key). Such tokens where $|K| \geq v$ are Efficient because by transmitting only vL_{p} bits, the token is able to release useful information amounting to $|K|L_{\mathrm{p}}$ bits which is greater than or equal to the

amount of bits transmitted.

Inefficient Tokens: Tokens which release less than v instances of one or more keys are termed Inefficient ($|K| \leq v$). The reasons for their inefficiency are obvious from the previous discussion on Efficient Tokens. Inefficient tokens waste bandwidth because most partitions are filled with random numbers which only ensure security and exclusive key release. If a token is heavily underutilized, then transmitting the keys through other key distribution mechanisms could save bandwidth. The process of assigning keys to different groups plays an important role in minimizing the number of Inefficient tokens. For an arbitrary configuration with a large number of nodes, the number of inefficient tokens is generally small. Usually, the final few tokens that are transmitted to complete a given configuration, are inefficient.

8.2 Comparisons with Tree based Broadcast encryption

Tree based broadcast encryption schemes impose a hierarchical structure on the network and inherently form groups by doing so. Efficient low memory tree based schemes require each node to store $O(logn)$ keys. If the controller decides to form N_{G} multicast groups in the network, the group key establishment phase would require the transmission of only $O(N_{\mathrm{G}})$ messages provided these groups are the same as the inherently formed ones. However, if arbitrary multicast groups are required, $O(m)$ messages would have to be transmitted for each group of average size m. This implies that formation of N_{G} arbitrary groups would require the transmission of $O(mN_{\mathrm{G}})$ messages. On the other hand, with the proposed scheme, any arbitrary multicast group with a size m greater than v would necessarily have overlapping subsets. Hence, in general less than m messages would be required to establish the group key. Moreover, the multicast groups can grow simultaneously as well. Assuming each token adds k nodes to each group, one token would release keys in kN_{G} nodes on an average. The total number of messages required would be $O(m/k)$ which is much lesser when compared to that of the Tree based Broadcast encryption schemes. However, for a network of size v^2, each node would have to store $O(v)$ keys which is higher than the storage requirement of low memory broadcast encryption methods.

8.3 Forward secrecy

Nodes which leave a multicast group should not have acess to future conversations. This can be achieved by creating one additional key not known to the leaving member of the group. Fast group key revocation with minimum number of messages requires the group members to form valid subsets in multiple key grids thereby imposing additional constraints on the design. Furthermore, if the design is changed to include such partially overlapping subsets to form one specific multicast group, it could reduce the flexibility in forming other groups linked to the same key. Consider a scenario of a network with four nodes 1, 2, 3 and 4. Now let keys K_1, K_2 and K_3 be released such that the following multicast group associations are formed : Now

Table 8.1: Initial group associations

Group	**Group key**
$G_1 = \{1,2,3\}$	K_1
$G_2 = \{1,3,4\}$	K_2
$G_3 = \{2,3,4\}$	K_3

assume node 1 leaves the network. This would imply that node 1 should no longer have the priviledge to future conversations within groups G_1 and G_2. This means that nodes 2 and 3 should share a unique common multicast key while the same is true for nodes 3 and 4. Individual nodes could form all possible functional combinations of keys released in them. This would imply the mapping shown in Table. 8.2.

Table 8.2: Initial Node - key associations

Node	**Revealed Keys**
1	K_1, K_2, H(K_1, K_2)
2	K_1, K_3, H(K_1, K_3)
3	K_1, K_2, K_3, All possible hashes
4	K_2, K_3, H(K_2, K_3)

Since node 1 leaves the network all keys unlocked in 1 cannot be used further. The corresponding mapping is shown in Table. 8.3.

The new group associations are given by $G_1` = \{2,3\}$, $G_2` = \{3,4\}$, $G_3` = \{2,3,4\}$ with group keys H(K_1, K_3), H(K_2, K_3) and K_3 respectively. Node 1 has been removed from Groups

Table 8.3: Node - key associations after revocation of node 1

Node	**Revealed Keys**
2	K_3, H(K_1, K_3)
3	K_2,K_3, H(K_2,K_3), H(K_1,K_3), H(K_1,K_2,K_3)
4	K_3, H(K_2,K_3)

G_1 and G_2 without the release of any new keys. The other members of each group still continue to be a part of their respective groups.

However, if node 3 decides to leave all its groups, then all the keys in the other three nodes would have to be updated which is in stark contrast to the previous scenario where no new keys were released. In some situations it is also possible to use even the compromised keys to create group keys for the modified groups without revealing any information about the new group keys to the leaving node.

Another possible solution is to form all possible multicast group associations with a unique key for each association.This would mean that the keys corresponding to subsets which contain the leaving node can be discarded and other keys can be used. This pre-designed approach would impose constraints on flexibility of group associations in the network and increase the number of pre defined distinct keys K_{i}.

We can define depth as the total number of nodes which have left the network or their groups. This would imply that depth can vary from 1 to $n - 1$ for multicast group associations. Future work should revolve around minimizing the number of disctinct keys, messages and support forward secrecy at different depth values.

Chapter 9

Simulations

We analyse the performance of the scheme based on the number of tokens that have to be transmitted to configure the network. We study a scenario where the system contains a fixed number of nodes and we find the number of tokens required to form all possible multicast groups of different sizes. It must be noted here that the sizes of the networks simulated are small and as a result, the number of required tokens should tend to be small as well. Consequently, significant bandwidth gains cannot be expected because the final few tokens that are transmitted, often tend to be inefficient. However, in larger networks, most of the transmitted tokens are likely to be Efficient and wastage of bandwidth over all tokens is less likely. A seven key, five node network is used in both scenarios. Since the network has 5 nodes, there are ten groups of size 2, ten of size 3 and five groups of size 4. The five node share code words are generated according to the procedure described in section 3.5. Further, it is assumed that every node computes and stores some function or hash of all possible combinations of keys that were unlocked in that node.

Group size 2: By unlocking keys according to the following table, a unique group key can be established for every possible group of size 2. The algorithm described in section 4 was used to construct tokens and we found that all ten groups, each of size 2, can be created by the transmission of four broadcast tokens.

Group size 3: For every group of size 3, a unique group key can be obtained as a function of two primary keys K_{i} and K_{j} for some i and j. Table 7 shows the key mappings for every possible triad of nodes. Similar to the previous case, we found that all the ten triads can be created by transmitting 4 tokens.

Table 9.1: Multicast groups of size 2

Group	**Group key**	**Group**	**Group key**
{1,2}	H(K_1,K_2)	{2,4}	H(K_1,K_6)
{1,3}	H(K_1,K_3)	{2,5}	H(K_2,K_5)
{1,4}	H(K_1,K_4)	{3,4}	H(K_1,K_7)
{1,5}	H(K_2,K_4)	{3,5}	H(K_5,K_7)
{2,3}	H(K_1,K_5)	{4,5}	H(K_4,K_7)

Table 9.2: Multicast groups of size 3

Group	**Group key**	**Group**	**Group key**
{1,2,3}	H(K_1,K_2)	{1,4,5}	H(K_3,K_4)
{1,2,4}	H(K_1,K_3)	{2,3,4}	H(K_1,K_5)
{1,2,5}	H(K_2,K_3)	{2,3,5}	H(K_2,K_5)
{1,3,4}	H(K_1,K_4)	{2,4,5}	H(K_3,K_5)
{1,3,5}	H(K_2,K_4)	{3,4,5}	H(K_4,K_5)

Group size 4: The key-group mappings shown in Table 8, when implemented yield five groups after the broadcast of four tokens (Similar to the previous two cases).

Table 9.3: Multicast groups of size 4

Group	**Group key**
{1,2,3,4}	K_1
{1,2,3,5}	K_2
{1,2,4,5}	K_3
{1,3,4,5}	K_4
{2,3,4,5}	K_5

The simulations reveal that only 4 tokens were required to form as large as 10 groups signalling improvement over traditional static key establlishment schemes and broadcast encryption schemes.

Chapter 10

Conclusion

In this thesis report we have presented a design methodology for controlling the release of protected encryption keys in different nodes, to enforce a dynamic re-configuration of the virtual wireless network, with the help of broadcast tokens. With the proposed node share and token design, any arbitrary virtual multicast configuration can be realized through a sequence of carefully designed tokens. Both the node shares and the broadcast tokens are collusion resistant. Further extensions have been discussed, to incorporate forward secrecy, which comes with a price of compromising on the network flexibility. A paper on our proposal has been submitted to the ACM WiSEC 2014 conference in manchester, UK and it is pending approval.

Appendices

Appendix I

Unique mapping property

The following proof demonstrates that when the general node share code generation procedure described in section 3.5 is followed, the resulting node share codewords satisfy the the Unique mapping property. Consider any two node shares which are distributed to two nodes i and j. They are characterised by the tuples (q_1, r_1) and (q_2, r_2) respectively. It must be noted that the two tuples cannot be equal because of the fact that they represent two distinct nodes. Hence, at least one of the parameters, either q or r must be different.

Case 1: Assuming that the parameter r is different but q remains the same between the two tuples, then the corresponding code word halves are $C_2(r_1)$ and $C_2(r_2)$ respectively. But $C_2(r_2)$ is obtained from $C_2(r_1)$ by a certain number of circular shifts less than or equal to $v - 1$ (v circular shifts will result in the same sequence again). Since all the assigned key numbers (in the code word representation) are distinct in both $C_2(r_1)$ and $C_2(r_2)$, no column will have the same number as both entries when $C_2(r_1)$ and $C_2(r_2)$ are stacked one below the other. Hence the y coordinate of every key (partition of the second half of every key) will differ in the Nodes i and j.

Case 2: The parameter q is different but r remains the same between the two tuples. The argument is similar to the one presented in Case 1

Case 3: Both q and r are different, follows trivially from Case 1 and Case 2

Appendix II

Simultaneous key release

Let $S = \{(x_1, y_1),\ (x_2, y_2), \ldots,\ (x_{\mathrm{N}}, y_{\mathrm{N}}\}$ be a $valid\ \ subset$ of nodes in which a key say K_{i} has to be released. Then S is defined as the privileged set of key K_{i}. The following proof demonstrates that the Token design procedure described in section 3.7 indeed unlocks the key K_{i} in every node belonging to the privileged set S. Further, the proof also shows that any node which does not belong to the privileged set cannot obtain any information about the key K_{i}. Let T be a token that is generated using the token design procedure described in section 3.7 to unlock K_{i} exclusively in the privileged set S.

II.1 At a node belonging to the privileged set

Let the node j belong to the privileged set. Let it be represented by the point (x, y) with respect to key K_{i}. Since the token was randomly initialised, every other partition apart from partitions x and y contain random numbers. No other key (apart from K_1) could be unlocked at this node because of invalid stack relations. Recall that in the node share N_{j} K_{i1} is filled in the partition x and K_{i2} is filled in the partition y. Since both K_{i1} and K_{i2} were assigned the same key table number, they share the same stack equation in N_{i}. However, one of the partitions (either x or y) in the token T is filled with the complement of the corresponding key half. Fusion of the token with N_{j} results in unique stack equations for K_{i1} and K_{i2}. Hence both K_{i1} and K_{i2} can be extracted and concatenated to obtain key K_{i}

II.2 At a node which does not belong to the privileged set

Consider a node A which does not belong to the privileged set. Let it be represented by the point (x_a, y_a) with respect to key K_i. Let us assume that the x coordinate be the same for all nodes in the privileged set S i.e, $S = \{(x_k, y_1), (x_k, y_2), \ldots, (x_k, y_N\}$ where the x coordinate equals x_k.

Case 1 : $x_a = x_k$

In this case the y coordinate of A must be different from all the y_js of the privileged set S. This is because if $y_a = y^j$ for some y_j in the privileged set S, then the point (x_a, y_a) will lie inside the privileged set S which is not possible.

Hence, $y_a \neq y_j \, \forall \, j = 1, 2, \ldots, N$

If the token design procedure is followed, then the partition corresponding to y_a will be filled with a random number which is not equal to ${K_{i2}}^c$. Let this random number be K_R. Since $K_R \neq {K_{i2}}^c$, some bits of K_R must be the same as those of ${K_{i2}}^c$. Let these positions in the partition be represented by the set B. Since the partition corresponding to y_a is filled with a random number K_R , the stack equations would become invalid and the second half K_{i2} cannot be unlocked in node A. When node A tries to unlock K_{i1}, the bits corresponding to the partition x_a i.e P_{X_a} are released along with the bits present in the positions denoted by the set B. In other words when the node A tries to unlock K_{i1}, it instead gets $\{P_{X_a} \bigcup B\}$ i.e, a union of both these sets. It cannot determine the partition P_{X_a} from this union. Hence K_{i1} would not be unlocked. Thus both halves K_{i1} and K_{i2} cannot be unlocked. No other key would be unlocked as well because every other partition would have invalid stack relations.

Case 2 : $x_a \neq x_k \; and \; y_a = y_j \; for \; some \; y_j \; in \; S$

Using similar arguments as in Case 1, it can be clearly seen that the partition corresponding to x_a i.e P_{X_a} will be filled with some random number K_{R1} . This will ensure that K_{i1} cannot be unlocked by node A. Since $K_{R1} \neq K_{i1}$ some bits in K_{R1} will be equal to those in ${K_{i1}}^c$. These will again form a non-null set B which will ensure that K_{i2} is not unlocked. (Similar to Case 1). No other key would be unlocked as well because every other partition would have invalid stack relations.

Case 3 : $x_a \neq x_k \; and \; y_a \neq y_j \; \forall \; j = 1, 2, \ldots, N$

In this case, both the partitions corresponding to x_{a} and y_{a} will be filled with random numbers. This yields invalid stack equations and both halves cannot be unlocked by node A. No other key would be unlocked as well because every other partition would have invalid stack relations.

No other cases are possible. This proves that the key K_{i} is not unlocked in any other node which does not belong to the privileged set. Similar arguments can be given when for the scenario where all the y coordinates of the privileged set S are equal and all the x coordinates are distinct.

Bibliography

[1] K. Karthik. Virtually reconfigurable secure wireless networks using broadcast tokens. *International Conference on Network and System Security. Springer Berlin Heidelberg, pp 599-606, 2013.*

[2] K. Karthik and D. Hatzinakos. Multimedia encoding for access control with traitor tracing: Balancing secrecy, privacy and traceability. *AV Akademikerverlag, ISBN: 978-3639452976, 2012.*

[3] J. Lee and D. R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. *IEEE Wireless Communications and Networking Conference, 2005.*

[4] J. Lee and D. R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. *Selected Areas in Cryptography. Springer Berlin Heidelberg, pp 294-307, 2005.*

[5] P. N. Liu, Donggang and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security (TISSEC), vol. 8, no. 1, pp 41-77, 2005.*

[6] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks. ACM, 2003.*

[7] R. Blom. An optimal class of symmetric key generation systems. *Advances in cryptology. Springer Berlin Heidelberg, 1985.*

[8] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Information and Computation,* 146(1):1 - 23, 1998.

[9] A. Fiat and M. Naor. Broadcast encryption. *Advances in Cryptology CRYPTO 93. Springer Berlin Heidelberg, 1994.*

[10] N. Kogan, Y. Shavitt, and A. Wool. A practical revocation scheme for broadcast encryption using smart cards. *ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 3, pp 325-351, 2006.*

[11] K. Karthik. A ramp code for fine-grained access control. *Intl. Conf. on Computer Science and Information Technology, CCSIT. 2013.*

Acknowledgments

We would like to thank our guide Professor Kannan Karthik for his valuable inputs which were instrumental in formulating our proposal. We would also like to thank our friends for their encouragement and support. We would like to reserve special acknowledgement for Rahul Nallamothu and Mridul Krishna for their useful ideas during the problem formulation. We also wish to acknowledge our parents who motivated and supported us throughout the project.

Date: _____

Atishay Jain , Vignesh Babu