BACHELOR

Minimizing the number of keys for secure communication in a network with colluders

Duif, N.

Award date:
2009

Minimizing the number of keys for secure communication in a network with colluders

Author: Niels Duif
Supervisor: Prof. dr. ir. Henk van Tilborg

Technische Universiteit Eindhoven

November 2, 2009

Student number: 0554878
Subject code: 2J008

Abstract

To allow participants in a network to communicate securely, symmetric cryptography may be used. But it is inefficient to assign a unique key to each pair of participants. Under the assumption that some of the participants may be trusted, secure communication is possible using fewer keys. It is assumed that at most t participants collaborate to retrieve messages that are sent through the network. Desmedt et al. show how to minimize the number of keys per participant under this assumption [1]. However, in some applications the *total* number of keys in the network is a more important parameter. This paper focuses on minimizing the total number of keys, while secure communication is maintained.

This paper presents an optimal construction for the case where there is at most 1 corrupt participant, that is $t = 1$. It also presents constructions for $t > 1$ based on block designs. The best constructions achieve a total number of keys of $O(t \cdot \sqrt{n})$ for n participants, but only for specific values of n. Therefore, a way of combining constructions is presented. In this way constructions for any number of participants n may be obtained. It is conjectured that in this way the total number of keys is at most $O(t^2 \cdot \sqrt{n})$.

Contents

Chapter 1

Introduction

This paper deals with secure communication in a network. An example is a computer network in which two of the participants want to send each other messages in such a way that no other participant may reconstruct them. Participants will be called the nodes of the network N, and are denoted by $N_1, N_2, \ldots, N_n$. So there are n participants in total. It is assumed that the network is connected, so that every pair of nodes can communicate directly or indirectly. A way to facilitate secure communication is to give each pair of nodes $\{N_i, N_j\}$ a unique key k_{ij}. This key is used for symmetric cryptography. Node N_i can then encrypt his message M with the key k_{ij} and send it to N_j who decrypts it. No other node can reconstruct the message, provided that the encryption method is secure. From now on, it is assumed that an encryption method without any known weaknesses is used.

However, the described method is not very efficient. In a network with n nodes the construction described above requires $\binom{n}{2} = \frac{n(n-1)}{2}$ keys in total. This report describes methods that use fewer keys, but still allow safe communication. For this it is assumed that at least some of the nodes are not trying to intercept and decrypt messages that are not intended for them. In particular, at most t nodes are expected to be corrupted and possibly working together. It is shown that with this assumption significantly fewer keys are needed.

This chapter introduces the notation for a message, a network, nodes, colluders, a key space, key sets, and keys. Section 1.2 gives an example of safe communication in a network with 4 nodes of which at most 1 is corrupted. Section 1.4 introduces Harary graphs, which were used by Desmedt, Van Tilborg, and Wang for secure communication [1]. It also shows an improvement on a construction that uses such a Harary graph.

Chapter 2 gives an optimal construction for the case in which there is only one colluder, $t = 1$.

Chapter 3 gives constructions for more than 1 colluder, $t > 1$. Section 3.1 derives a theoretical lower bound on the total number of keys, which is not met by the constructions described in Section 3.2. Section 3.2 does not give constructions for all values of n, but Chapter 4 shows how multiple constructions may be combined to obtain constructions for any number of nodes, n. The constructions presented in 3.2 were also published in [4].

Chapter 5 gives the conclusions of this paper, and some recommendations for future research.

1.1 Definitions

The network N is connected and consists of n nodes. This means that n people can communicate with each other. It is assumed that the protocol of the network is such that any person in the network may intercept all data that are transmitted. So the physical topology of the network is not important. Many local area networks use such a protocol. This means the sent data must be encrypted to allow private communication. The number of colluders in the network is assumed to be no greater than t. In other words, at most t nodes are corrupted. So at most t nodes work together in eavesdropping on some or all communication. All nodes, including colluders, are expected to obey to the protocol.

The nodes will be labelled 1 to n and called $N_1, N_2, \ldots, N_n$. Sometimes they are indicated by capital letters starting from A instead. Each node N_i has a set of keys K_i. A key set contains keys from the key space K. The key space has a total number of c keys that are labelled $k_1, k_2, \ldots, k_c$. This means that each K_i is a subset of K. Sometimes the keys will just be denoted by their numbers. The colluders will be called $E_1, E_2, \ldots, E_t$. The number of colluders is always assumed to be t. If safe communication is possible in a network prone to t colluders, safe communication is also possible in the presence of fewer colluders. A network that allows safe communication in the presence of t colluders is called t-safe. The message to be sent will be called M.

Definition 1.1.1. *A key assignment that is safe in the presence of at most t colluders is called t-safe.*

This means that a t-safe key assignment is also $(t-1)$-safe.
To avoid degenerate cases it is assumed that every key is shared between at least two nodes. This is a reasonable assumption, because a key that is not shared between at least two nodes can only be used for encryption and decryption by a single node. Such a key is useless for communicative purposes.

1.2 Example of a safe communication network

The network in Figure 1.1 has $n = 4$ nodes. The key space $K = \{1, 2, 3, 4\}$ has 4 keys. At each node N_i the key set K_i is given. For example, node N_1 has keys k_1 and k_2, which means $K_1 = \{1, 2\}$. All nodes that share one or more keys are connected by a line. The label beside the line shows the common keys.
If N_1 wants to send a message to N_2, he simply uses their common key k_2. Since N_3 and N_4 do not know k_2 they cannot decrypt this communication.

If N_1 wants to send a message to N_3 the sender and receiver have no key in common. N_1 can send his message M to N_2 using key k_2. N_2 can then send it to N_3 using k_3. But this communication is not safe since N_2 can eavesdrop. So N_1 decides to split up his message as follows: $M = M_1 \bigoplus M_2$. Here " $\bigoplus$ " denotes bitwise addition modulo 2, which is also known as the XOR-function. M_1 is chosen at random and M_2 is computed by $M_2 = M \bigoplus M_1$. Indeed $M_1 \bigoplus M_2 = M_1 \bigoplus M \bigoplus M_1 = M$. M_1 is sent through N_2 to N_3 using keys k_2 and k_3. M_2 is sent through N_4 using k_1 and k_4. Now N_2 can only determine M_1 since he does not have k_1 or k_4. Similarly N_4 can only determine M_2. M_1 and M_2 give no information about the original message M. So to eavesdrop on this conversation N_2 and N_4 must work together. This means that the network allows safe communication in the presence of at most 1 colluder, so $t = 1$.

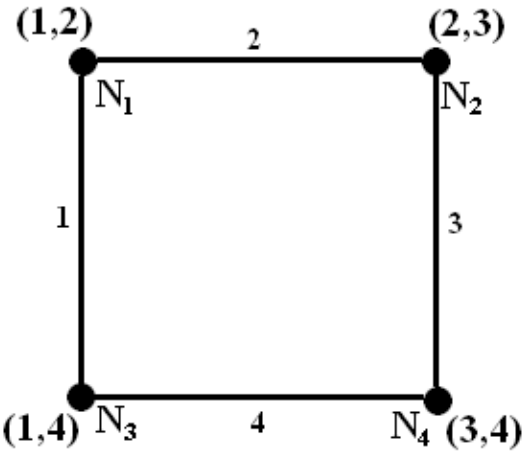


Figure 1.1: In this network each node has two keys that are given next to the node. Nodes are connected if they share one or more keys. The common keys are displayed next to the connections. Safe communication is possible in the presence of at most $t = 1$ colluder.

The idea of splitting up a message M as the XOR of random shares, plays a crucial role in this paper. In such a splitting all shares but one are determined at random. The last share is such that the XOR of all shares is indeed M.

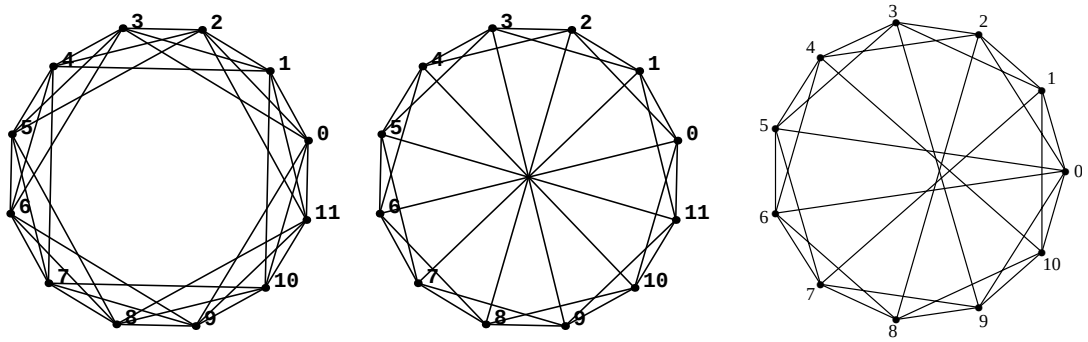
Construction 1.2.1. *A message M can be split up into s shares by using the XOR function. The XOR function is bitwise addition modulo 2, and is denoted by "$\bigoplus$". The splitting of a message into s shares is performed as follows. $s-1$ shares, $M_1, M_2, \ldots, M_{s-1}$ are determined at random. The share M_s is computed as $M_s = M \bigoplus M_1 \bigoplus M_2 \bigoplus \cdots \bigoplus M_{s-1}$. Then the XOR of all shares $M_1, M_2, \ldots, M_S$ is indeed M, because $M_i \bigoplus M_i = 0$ holds for any bit string. Furthermore, any set S of at most $s-1$ shares gives no information on M. This is because the XOR of M with all elements of S is the XOR of $s - |S|$ random bit strings. This means that, given S, all possible bit strings for M have equal probability.*

1.3 Constructions using Harary graphs

An existing construction that enables safe communication in a network with n nodes and at most t colluders, uses *Harary graphs* [4]. A Harary graph is defined on n nodes, having *connectivity* h. The connectivity is not important in this paper, so it is not discussed.

The example in Figure 1.2 shows Harary graphs for $n = 11, h = 5$, $n = 12, h = 5$, and $n = 12, h = 6$. [4] gives the following definition of a Harary graph.

Definition 1.3.1. *The Harary graph $H_{n,h}$, $n \geq h+1$, is the graph on n vertices, numbered $0, 1, \ldots, n-1$, with calculations modulo n, where vertices i and j, $j \neq i$, are connected if and*



(a) $n = 12, h = 6$ (b) $n = 12, h = 5$ (c) $n = 11, h = 5$

Figure 1.2: The three different types of Harary graphs. In Figure (a) h is even, in Figure (b) n is even and h is odd, and in Figure (c) n and h are both odd. All connected nodes share a unique common key.

only if

$$|j-i|_n \leq l \quad \text{if} \quad h = 2l, \tag{1.1}$$

$$|j-i|_n \leq l \ \text{ or } \ |j-i|_n = m \quad \text{if} \quad h = 2l+1 \ \text{ and } \ n = 2m, \tag{1.2}$$

$$|j-i|_n \leq l \ \text{ or } \ j = i+m+1, i \in \{0,1,\ldots,m\} \quad \text{if} \quad h = 2l+1 \ \text{ and } \ n = 2m+1, \tag{1.3}$$

where $|j-1|_n$ is the shortest distance between i and j along the modular circle.

The nodes in a Harary graph are arranged in a circular pattern. The graph is transformed into a key assignment scheme by assigning a unique key to every edge of the graph. So every edge represents a key that is only shared by the two vertices it connects. A key with this property will be called 'uniquely shared'. [4] shows that a key assignment that uses a Harary graph with connectivity h, is $(h-1)$-safe, so $h = t+1$. The communication is secured by splitting the message M into $t+1$ shares that are sent over $t+1$ vertex disjoint paths. The splitting is done by Construction 1.2.1. In a Harary graph $H_{n,h}$ it is always possible to find h different vertex disjoint pahts. A proof of this claim can be found in [4]. Here only an example is given. In Figure 1.2(b) the following 5 paths between node 0 and node 5 are vertex disjoint: $0 \to 2 \to 4 \to 5$, $0 \to 1 \to 3 \to 5$, $0 \to 10 \to 8 \to 7 \to 5$, $0 \to 11 \to 5$, and $0 \to 6 \to 5$. This means that an adversary of $t = 4$ nodes is not sufficient to determine all shares of the message M, unless it contains node 0 or node 5.

If $t+1$ is even, each node gets $t+1$ keys, which are shared with the nearest nodes as measured along the modular circle. In the other two cases, shown in Figure 1.2(b) and 1.2(c), the 'diagonals' or 'skew diagonals' are added to connect the antipodal nodes. Note that if n and $t+1$ are both odd, as in Figure 1.2(c), node 0 gets $t+2$ keys instead of $t+1$. In all other cases, each node gets $t+1$ keys, so a total number of $c = \frac{(t+1)\cdot n}{2}$ keys is needed. This means that in general these constructions use at most $c = \lceil \frac{(t+1)\cdot n}{2} \rceil$ keys. If n and $t+1$ are both odd, sometimes one fewer key is needed, as is shown in [4].

1.4 Improvement on existing constructions

In most cases a construction that uses a Harary graph minimizes the number of keys per node. However, constructions exist that use a smaller *total* number of keys than Harary graphs. The example in Figure 1.3 shows a Harary graph for $n = 5$ and $t = 2$. It is drawn in a different way from the examples in Figure 1.2 to prevent crossing lines. It is easy to see that the graph is 2-safe, since between any two nodes there are at least three vertex-disjoint paths. Surprisingly, if key 8 is replaced by key 6 the graph still is 2-safe. To prove this, it is sufficient to show that safe communication is possible from B to E and from B to D. All other communication routes remain unchanged or are isomorphic to one of these cases. Communication from B to E: since B and E do not share a unique key anymore, B uses the routes B $\xrightarrow{6}$ E, B $\xrightarrow{2}$ C $\xrightarrow{7}$ E, and B $\xrightarrow{1}$ A $\xrightarrow{5}$ E. Now D can determine M_1, which was sent from B to E using k_6. But he cannot determine M_2 or M_3 since none of his keys is used for their encryption. If D is a collaborator the communication cannot be intercepted because M_2 and M_3 are sent using vertex-disjoint paths and uniquely shared keys. If D is not a collaborator, the case is similar to the original Harary graph. Communication from B to D: this communication uses the same routes as in the original case: B $\xrightarrow{1}$ A $\xrightarrow{4}$ D, B $\xrightarrow{6}$ E $\xrightarrow{6}$ D, and B $\xrightarrow{2}$ C $\xrightarrow{3}$ D. The only part of the route that could be unsafe is B $\xrightarrow{6}$ E $\xrightarrow{6}$ D, because the rest is similar to the original Harary Graph. The only node that knows k_6 besides B and D is E. E gets to know the message encrypted by k_6 in the original Haray graph too, so the security is not compromised.

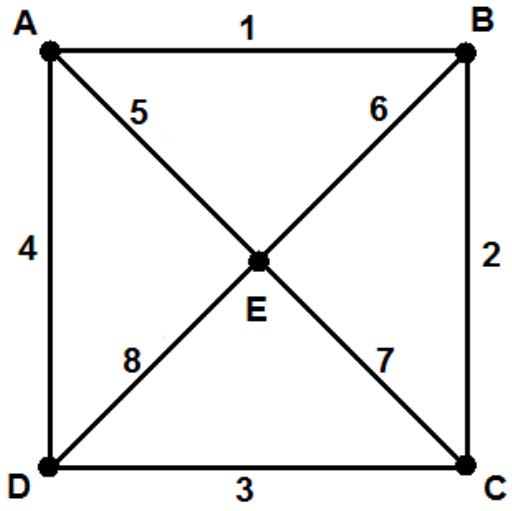


Figure 1.3: The Harary Graph for $n = 5$ and $t = 2$ is 2-safe. If the key k_8 that is shared between D and E is replaced by key k_6 the graph is still 2-safe.

Chapter 2

Constructions for t=1

In this chapter it is assumed that t=1. This means that there is one corrupted node in the network. It is assumed that $n > 2$. A message M is split into s pieces: $M = M_1 \bigoplus M_2 \bigoplus \cdots \bigoplus M_s$. To allow communication the receiver must be able to deccrypt all s pieces. On the other hand a colluder must not be able to intercept all parts M_i of the message M. So a necessary condition is:

Lemma 2.0.1. *One node cannot possess all keys of another node:*

$$\forall_{i \neq j} K_i \not\subseteq K_j. \tag{2.1}$$

Proof. Assume $K_i \subseteq K_j$ for some $i \neq j$. Then N_j possesses all key of N_i. This means that N_j can eavesdrop on all communication from N_i. This is not allowed unless $t = 0$ or $n \leq 2$. □

2.1 Lower bound

To obtain a lower bound for the total number of keys c, it is assumed that each node N_i is assigned the same number of keys. So $|K_i| = k \ \forall\, i$. It will be shown that this is an optimal choice. By Lemma 2.0.1 no two nodes can have the same key set. If the total number of keys is c, this means that the number of nodes cannot exceed $\binom{c}{k}$. So the maximum number of nodes n given a total number of keys c equals

$$n = \max_k \binom{c}{k} = \binom{c}{\lfloor \frac{c}{2} \rfloor}. \tag{2.2}$$

This equality follows from the following theorem.

Theorem 2.1.1. *The maximum cardinality of a collection of subsets of a t-element set T, none of which contains another, is the binomial coefficient $\binom{t}{\lfloor \frac{t}{2} \rfloor}$.*

This theorem was proved by Emanuel Sperner [2] in 1928. The theorem is sometimes referred to as Sperner's Lemma, but that name also refers to another result by Sperner on graph colouring. Theorem 2.1.1 also shows that the choice $|K_i| = k \ \forall\, i$, is indeed optimal.

2.2 An optimal construction for t=1

This section shows the existence of networks that meet Equation (2.2) with equality. To achieve this, consider the key space K of size $|K| = c$. In this key space all subsets of size $\lfloor \frac{c}{2} \rfloor$ are taken and distributed among $n = \binom{c}{\lfloor \frac{c}{2} \rfloor}$ nodes. This key assignment meets Equation (2.2) with equality.

First, an example is given that uses such a key assignment scheme. Then it is proved that such a key assignment can be used for secure communication in the presence of $t = 1$ colluder. Table 2.1 shows a key assignment scheme with $c = 4$ keys. The $n = \binom{4}{\lfloor \frac{4}{2} \rfloor} = 6$ nodes possess all 6 different 2-sets of the key space.

	key			
node	1	2	3	4
A	x	x		
B	x		x	
C	x			x
D		x	x	
E		x		x
F			x	x

Table 2.1: This key assignment scheme uses all different 2-sets from a key space with 4 keys.

Suppose that in the case of Table 2.1 node A wants to send a message M to node F. Node A first splits up his message $M = M_1 \bigoplus M_2 \bigoplus M_3 \bigoplus M_4$. He then determines all combinations of A's keys and F's keys. These are {1,3}, {1,4}, {2,3}, and {2,4}. M_1 is sent using the first combination of keys: {1,3}. A encrypts M_1 with his key k_1 and sents it to the node that knows the combination {1,3}. If $c \geq 4$ such a node always exists. In this case it is node B. Node B decrypts M_1 and sends it on to F encrypted by key k_3. The messages M_2, M_3, and M_4 are sent using the other key combinations, so they are sent through node C, D, and E respectively.

Theorem 2.2.1. *The communication graph on $\binom{c}{\lfloor \frac{c}{2} \rfloor}$ points with all possible $\lfloor \frac{c}{2} \rfloor$-size key sets is safe in the presence of 1 colluder if $c \geq 4$.*

Proof. The following construction will be proved safe in the presence of 1 colluder. Suppose node A want to send a message M to node B. Let d be the number of keys that A and B have in common. A splits the message M into two parts: $M = M_D \bigoplus M_I$. M_D is sent directly to B. It is encrypted by all common keys $k_1, k_2, \ldots, k_d$. If the cryptographic protocol does not support encryption by multiple keys then M_D is split into d shares that are sent separately, encrypted by $k_1, k_2, \ldots, k_d$ respectively. This means that a colluder needs the keys $k_1, k_2, \ldots, k_d$ to determine M_D.
A and B both have $\lfloor \frac{c}{2} \rfloor - d$ keys none of which are common keys. They are labelled $k_{A_{d+1}}, k_{A_{d+2}}, \ldots, k_{A_{\lfloor \frac{c}{2} \rfloor}}$, and $k_{B_{d+1}}, k_{B_{d+2}}, \ldots, k_{B_{\lfloor \frac{c}{2} \rfloor}}$ respectively. Now M_I is split up into $(\lfloor \frac{c}{2} \rfloor - d)^2$ shares: $M_I = M_{d+1,d+1} \bigoplus M_{d+1,d+2} \bigoplus \cdots \bigoplus M_{\lfloor \frac{c}{2} \rfloor, \lfloor \frac{c}{2} \rfloor}$. Now $M_{d+1,d+1}$ is sent encrypted with $k_{A_{d+1}}$ to a node that possesses both $k_{A_{d+1}}$ and $k_{B_{d+1}}$. This node exists because $c \geq 4$ and the key assignment uses all possible $\lfloor \frac{c}{2} \rfloor$-size key sets. That node decrypts $M_{d+1,d+1}$, encrypts it with $k_{B_{d+1}}$ and sends it on to B. All other shares of M_I are sent in the same way,

using all combinations of the keys that A and B do not have in common.
To determine M_I a colluder needs all shares $M_{d+1,d+1}, M_{d+1,d+2}, \ldots, M_{\lfloor \frac{c}{2} \rfloor, \lfloor \frac{c}{2} \rfloor}$. To determine the set of shares $M_{d+1,d+1}, M_{d+1,d+2}, \ldots, M_{d+1,\lfloor \frac{c}{2} \rfloor}$, a colluder either needs $k_{A_{d+1}}$ or the keys $k_{B_{d+1}}, k_{B_{d+2}}, \ldots, k_{B_{\lfloor \frac{c}{2} \rfloor}}$. The latter cannot occur, since the colluder then knows all of B's keys which contradicts Lemma 2.0.1. So the colluder knows $k_{A_{d+1}}$. By the same reasoning on the shares $M_{d+2,d+1}, M_{d+2,d+2}, \ldots, M_{d+2,\lfloor \frac{c}{2} \rfloor}$, the colluder needs to know $k_{A_{d+2}}$. This reasoning is continued to conclude that the colluder knows all of $k_{A_{d+1}}, k_{A_{d+2}}, \ldots, k_{A_{\lfloor \frac{c}{2} \rfloor}}$. But then he knows all of A's keys, which contradicts Lemma 2.0.1. So one colluder cannot determine $M = M_D \bigoplus M_I$. So the communication graph on $\binom{c}{\lfloor \frac{c}{2} \rfloor}$ points with all possible $\lfloor \frac{c}{2} \rfloor$-size key sets is safe in the presence of 1 colluder if $c \geq 4$. □

The total number of keys needed for the described key assignment turns out to be very small. Using Stirling's approximation and some rounding for odd c, the following result is found for the total number of keys c:

$$\begin{aligned} n &= \binom{c}{\lfloor \frac{c}{2} \rfloor} = \frac{c!}{(\lfloor \frac{c}{2} \rfloor)!^2} \approx \frac{\sqrt{2\pi c}\left(\frac{c}{e}\right)^c}{2\pi \frac{c}{2}\left(\frac{c}{2e}\right)^c} = \sqrt{\frac{2}{\pi}} \cdot 2^c \\ c &\approx \log_2(n) + \log_2\left(\sqrt{\frac{\pi}{2}}\right) \end{aligned} \tag{2.3}$$

Unfortunately such a key assignment does not exist for any number of nodes n. Table 2.2 shows all $n < 1000$ for which such a key assignment exists and it lists the corresponding total number of keys c.

Number of nodes, n	3	6	10	20	35	70	126	252	462	924
Total number of keys, c	3	4	5	6	7	8	9	10	11	12

Table 2.2: For these numbers of nodes $n < 1000$ an optimal key assignment exists for t=1. This table also lists the number of keys c required for such an assignment.

Chapter 4 shows how multiple block designs may be combined to get a key assignment scheme for any desired number of nodes.

The number of communication routes that is used, may become rather large. For communication between two nodes that share p keys the number of communication routes equals $(\lfloor \frac{c}{2} \rfloor - p)^2$. So the number of routes is at most $(\lfloor \frac{c}{2} \rfloor)^2$. For comparison: when each pair of nodes $\{N_i, N_j\}$ is given a unique key k_{ij} the number of communication routes is 1.

Chapter 3

Constructions for t>1

In this chapter it is assumed that $t > 1$. This means there are at least two colluders. When $t > 1$ a construction using $O(\log(n))$ keys is no longer possible. This chapter shows that at least $O(t \cdot \log(n))$ keys are needed. It gives constructions that use $O\left(t \cdot \sqrt{n}\right)$ keys.

In this chapter constructions for $t > 1$ are studied. To judge the asymptotic behaviour of a construction, the total number of nodes, n, is considered to be the most important parameter. In the next sections the number of colluders, t, is assumed constant, while the asymptotic behaviour of the total number of keys, c, is studied for increasing n. In Section 3.1 it is assumed that $n \gg t^2$ This means that the derived bound may only be good for very large n. However, it is shown that the predicted asymptotic behaviour of c already occurs for small values of n, that is as soon as $n > t$.

3.1 Lower bound

This section derives a lower bound for the total number of keys, c, that are needed. This number is derived for a t-safe non-degenerate key assignment, where t is constant. That the key assignment is non-degenerate means that every key is shared between at least two nodes. To derive a lower bound for the total number of keys needed, a simple counting argument is used. From the n nodes, at most t are corrupted. This means that any set of $a \leq t$ nodes may be an adversary. Lemma 3.1.1 shows that all possible adversaries must have different key sets.

Lemma 3.1.1. *All sets of at most t nodes must have different key sets.*

Proof. Assume two sets of nodes, A_1 and A_2, have the same key set, and that both sets are of size at most t. Also assume that $A_1 \neq A_2$. Then at least one of these sets, say A_1, contains a node N_1 that is not contained in A_2. Because the key sets are the same, A_2 can eavesdrop on all communication of A_1. But then A_2 can also eavesdrop on N_1, which contradicts the assumption that the key assignment is t-safe. □

There are $\binom{n}{a}$ possible adversaries of size a, and all of them have a different key set. Summing over all adversaries of size at most t, this gives the following bound:

$$2^c \geq \sum_{a=0}^{t} \binom{n}{a}. \tag{3.1}$$

The expression on the left is the total number of possible key sets. With the assumption $n \gg t$ the right hand side may be approximated as follows. For this approximation the *binary entropy* function is needed. This function is commonly used in information theory, and denoted by H.

$$H(x) := -x \log_2 x - (1-x) \log_2 (1-x), \qquad 0 < x \leq \frac{1}{2} \tag{3.2}$$

[7] gives the following estimate for the right hand side of Equation (3.1), which converges to an equality for large n:

$$\sum_{0 \leq a \leq \lambda n} \binom{n}{a} \approx 2^{nH(\lambda)}. \tag{3.3}$$

With $\lambda = \frac{t}{n}$, Equation (3.3) gives the following estimate for Equation (3.1):

$$\begin{aligned} 2^c &\gtrapprox 2^{nH(\lambda)} \\ c &\gtrapprox n\left(-\frac{t}{n}\log_2\frac{t}{n} - \left(1-\frac{t}{n}\right)\log_2\left(1-\frac{t}{n}\right)\right) = t\left(\log_2 n - \log_2 t\right) + (n-t)\left(\log_2\left(1-\frac{t}{n}\right)\right) \\ &\approx t\left(\log_2 n - \log_2 t\right) + (n-t)\cdot\left(\frac{t}{n\ln(2)} + o\left(\frac{t^2}{n^2}\right)\right) = t\log_2(n) + o(t\log(t)). \end{aligned} \tag{3.4}$$

The third line uses the Taylor series of $\log_2(1-\frac{t}{n})$, and the assumption $n \gg t^2$.

Even when n is not much larger than t, the number of keys c still increases approximately logarithmically in n and linearly in t. Figure 3.1 shows c as a function of n for different values of t. These values are found by determining the minimum values of c from Equation (3.1).

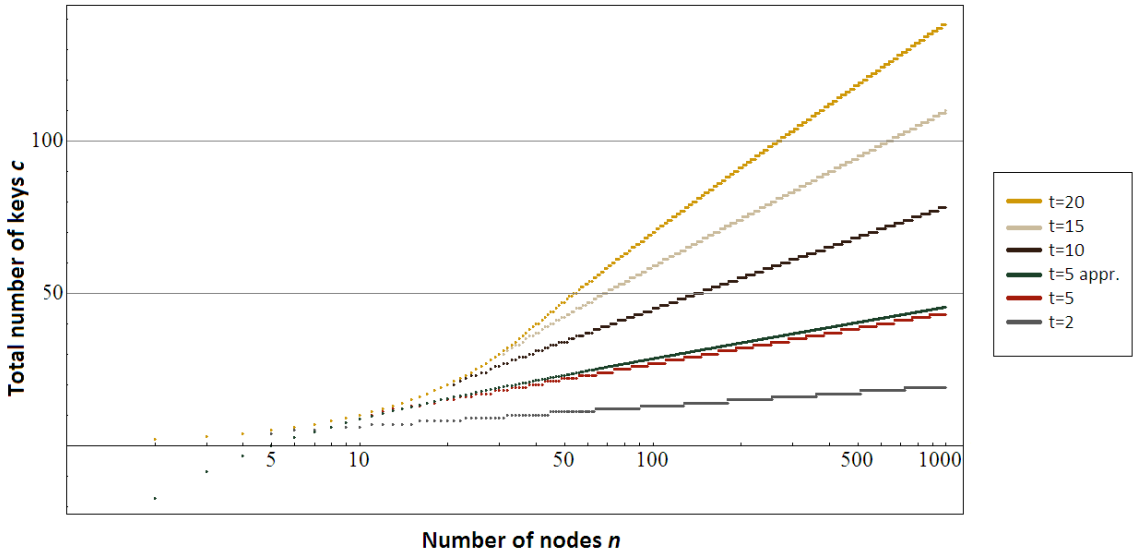


Figure 3.1: The minimum number of keys needed as a function of the number of nodes in a network. This lower bound is calculated with Equation (3.1).

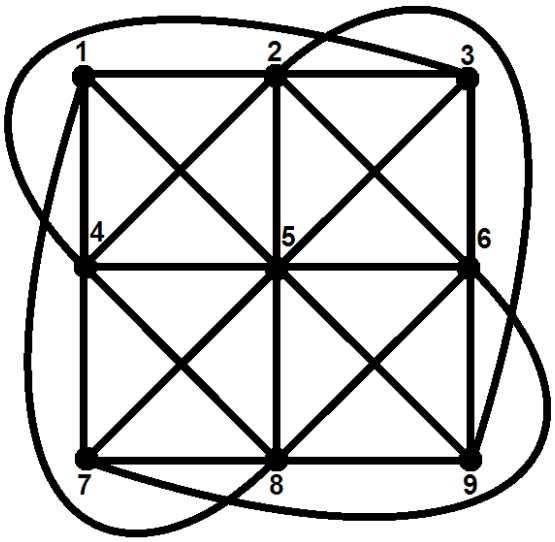


Figure 3.2: The affine plane over $\mathbb{F}_3$.

In Figure 3.1 the approximation for $t = 5$ uses $c \approx t\,(\log_2 n - \log_2 t) + \frac{t(n-t)}{n\ln(2)}$ from Equation (3.4).

3.2 Constructions using a 2-(v,t+1,1) design

In this section constructions are presented that use a $2-(v, t+1, 1)$ design. An example of such a design is the affine plane over $\mathbb{F}_3$, which is a 2-(9,3,1) design. This means it has 9 elements that are divided into subsets of 3 elements such that any 2 elements belong to a unique common subset. Figure 3.2 shows the affine plane over $\mathbb{F}_3$. The elements are the vertices, which are labelled 1 to 9. The subsets of 3 elements are the lines, which contain three points each. Some lines are curved but they always contain three points. For example, the curve through the points 4, 2, and 9 is also a line. As stated, any two points are connected by a unique line.
Many similar designs exist. These designs are called balanced incomplete block designs. Since only 2-designs are important for this paper, the definition of a 2-design is given. The full notation for such a design is a (v,b,r,k,λ) 2-design, which is denoted by B.

Definition 3.2.1. *A (v,b,r,k,λ) 2-design is defined on a set X of v elements called* points. *The 2-design B is a collection of b subsets of X called* blocks *such that every block contains k points, every point is contained in exactly r blocks, and the number of blocks that contain two given points x and y is always equal to λ. Since b and r are determined by the other parameters this is also called a 2-(v,k,λ) design.*

The *blocks* in Definition 3.2.1 will be called *lines* from now on. In this chapter a $2-(v, t+1, 1)$ design will be used. This means there are v points divided into $t+1$ subsets such that any two points are contained in a unique line. These designs only exist [3] when

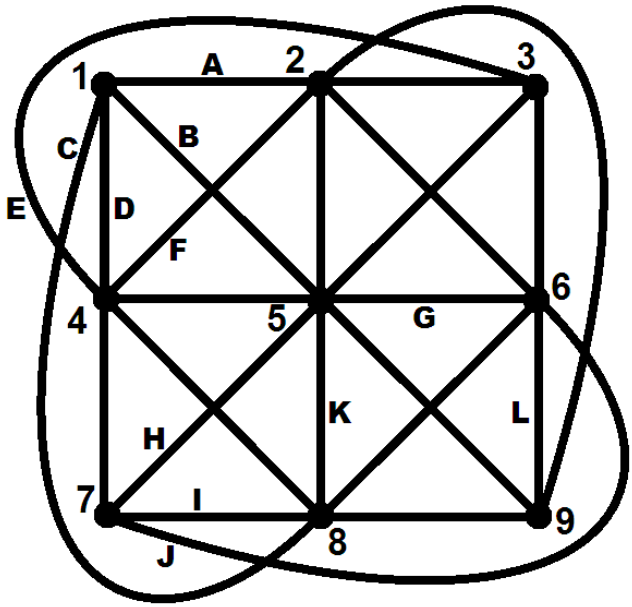


Figure 3.3: In a $2-(v,t+1,1)$ design the lines represent the nodes, and the points represent the keys. In this example the affine plane over $\mathbb{F}_3$ is used, which is a $2-(9,3,1)$ design. The nodes (lines) are labelled A to L, and the keys (points) are labelled 1 to 9.

$$v = \begin{cases} 1 \\ t+1 \end{cases} \pmod{t(t+1)}. \tag{3.5}$$

This is a necessary condition and not a sufficient condition. However when $t \leq 10$ few parameters v and t that satisfy (3.5) are known for which such a design does not exist [3]. When t is prime a $2-(t^2+t+1,t+1,1)$ design always exists. This is the projective plane of dimension 2 over $\mathbb{F}_t$. For now, n and t are supposed to be such that a $2-(v,t+1,1)$ design exists.

Now a construction is given that uses these designs. Consider the affine plane over $\mathbb{F}_3$ in Figure 3.3. The lines in this figure represent the nodes of the communication graph and the points represent the keys. Note that in the communication graphs in Chapter 1 nodes are represented by points and keys by lines! A point in Figure 3.3 represents a key that is shared by all lines through that point. For example, key k_6 is shared by node C, G, J, and L. And, for example, node B possesses keys k_1,k_5, and k_9. Every line passes through 3 points, so every node possesses $t+1=3$ keys.

Figure 3.3 can also be represented as a communication graph. Every point in Figure 3.3 represents a key. In a communication graph a key is represented by a line. Conversely, every line in Figure 3.3 represents a node. In a communication graph a node represented by a point. So the communication graph is obtained by replacing points with lines and replacing lines with points. The resulting structure is called the *dual* of the original structure.

The dual of Figure 3.3 that is obtained by interchanging lines and points, is given in

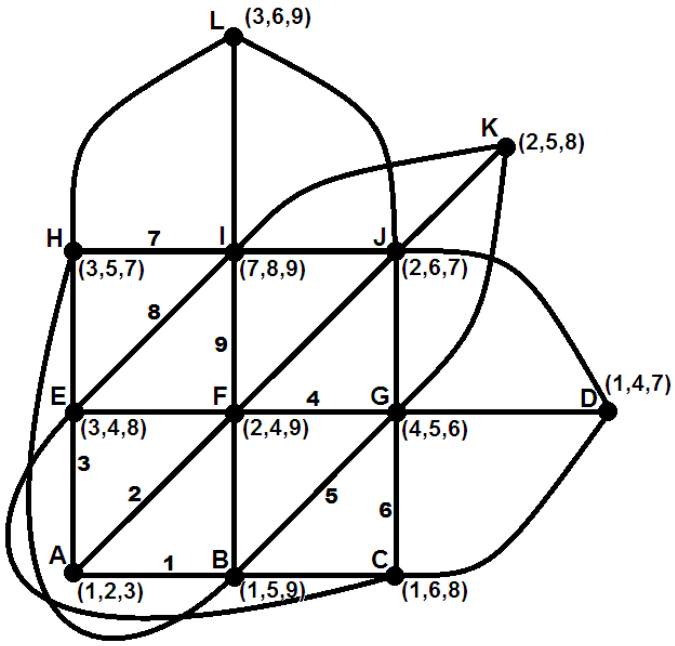


Figure 3.4: The dual of the affine plane over $\mathbb{F}_3$ can be used as a communication graph.

Figure 3.4. In Figure 3.4 the points represent the nodes and the lines represent the keys, as is usual in a communication graph.
To prove that the dual of a 2-$(v, t+1, 1)$ design is indeed a t-safe communication graph an important property of this dual is needed.

Lemma 3.2.2. *In the dual of a 2-$(v, t+1, 1)$ design every two lines intersect in a unique point.*

Proof. In a 2-design every two points are connected by a unique line. This means that in the dual every two lines intersect in a unique point. □

One more property of the dual of a 2-$(v, t+1, 1)$ design is needed.

Lemma 3.2.3. *In the communication graph that is the dual of a 2-$(v, t+1, 1)$ design, a set of t nodes cannot possess all keys of a node, A, unless that set contains node A.*

Proof. Assume the set of nodes $N_1, N_2, \ldots, N_t$ does not contain node A. Then each of $N_1, N_2, \ldots, N_t$ possesses at most one key that A also possesses; suppose N_i possesses k_1 and k_2 which are also possessed by A. By Lemma 3.2.2 N_i must be the same node as A. This contradicts the assumption that A is not in the set $N_1, N_2, \ldots, N_t$. So indeed each of $N_1, N_2, \ldots, N_t$ possesses at most one key that A also possesses. That means that the set $N_1, N_2, \ldots, N_t$ possesses at most t keys that A also possesses. A possesses $t+1$ keys so these are not all of A's keys. So a set of t nodes cannot possess all keys of node A. □

Now the main theorem of this section can be proved.

Theorem 3.2.4. *The dual of a 2-$(v, t+1, 1)$ design is a t-safe communication graph.*

Proof. To prove this statement, suppose that A and B are nodes in the communication graph. It is shown that safe communication is possible from A to B in the presence of at most t colluders. The proof splits into two cases.

Case 1: A and B have one or more keys in common
In this case A and B share exactly one key: suppose A and B have two keys in common, k_1 and k_2. By Lemma 3.2.2 A and B must be the same node. This contradicts the assumption that A and B are communicating nodes.

The unique key shared between A and B is k_1. This key is needed to eavesdrop on the communication. This requires one node E in the adversary. Now node E cannot possess another key of A or B: suppose w.l.o.g. that E possesses another of A's keys, k_2. Then E possesses k_1 and k_2. By Lemma 3.2.2 the point that possesses k_1 and k_2 is unique. So this point must be node A. This contradicts the assumption that E is in the adversary.

Now k_1 and node E can be deleted because E does not possess any other relevant keys. In the remaining construction A and B both have t keys and it must be shown that this construction is $(t-1)$-safe. This is similar to the proof of case 2 below.

Case 2: A and B have no common keys
Then A has keys $k_{A_1}, k_{A_2}, ..., k_{A_{t+1}}$ and B has keys $k_{B_1}, k_{B_2}, ..., k_{B_{t+1}}$. A splits up the message M into $(t+1)^2$ shares, $M = M_{1,1} \bigoplus M_{1,2} \bigoplus \cdots \bigoplus M_{t+1,t+1}$. For the protection of $M_{1,1}$ the keys k_{A_1} and K_{B_1} will be used. By Lemma 3.2.2 there is a unique node $N_{1,1}$ that possesses both of these keys, and since A and B have no common keys this node is not A or B. A sends $M_{1,1}$ to $N_{1,1}$ encrypted with k_{A_1}. $N_{1,1}$ decrypts it and sends it to B encrypted with k_{B_1}. B can then decrypt $M_{1,1}$. All other shares $M_{i,j}$ are sent in the same way. A sends $M_{i,j}$ to $N_{i,j}$ encrypted with k_{A_i}, where $N_{i,j}$ is the unique node that possesses both k_{A_i} and k_{B_j}. $N_{i,j}$ decrypts $M_{i,j}$ with k_{A_i}, encrypts it with k_{B_j}, and sends it to B. B then decrypts it with k_{B_j}. It is obvious that B can reconstruct the original message $M = M_{1,1} \bigoplus M_{1,2} \bigoplus \cdots \bigoplus M_{t+1,t+1}$. An adversary needs all shares $M_{1,1}, M_{1,2}, \ldots, M_{t+1,t+1}$ to determine M.

The only thing that remains to be shown, is that an adversary of size t or less cannot determine all shares $M_{1,1}, M_{1,2}, \ldots, M_{t+1,t+1}$.

To see this, consider the set of shares $M_{1,1}, M_{1,2}, \ldots, M_{1,t+1}$. All of these shares are first encrypted by k_{A_1}. Then they are sent on using all different keys $k_{B_1}, k_{B_2}, \ldots, k_{B_{t+1}}$. That means that to determine the shares $M_{1,1}, M_{1,2}, \ldots, M_{1,t+1}$, an adversary either needs k_{A_1} or the set of keys $k_{B_1}, k_{B_2}, \ldots, k_{B_{t+1}}$. The latter cannot happen by Lemma 3.2.3. So the adversary possesses k_{A_1}. The same observation on the set of shares $M_{2,1}, M_{2,2}, \ldots, M_{2,t+1}$ implies that the adversary must possess k_{A_2}. This argument is continued to conclude that the adversary must possess all of $k_{A_1}, k_{A_2}, \ldots, k_{A_{t+1}}$ to determine the shares $M_{1,1}, M_{1,2}, \ldots, M_{t+1,t+1}$. These are all of A's keys, which cannot happen by Lemma 3.2.3. So an adversary of size t or less cannot determine all shares $M_{1,1}, M_{1,2}, \ldots, M_{t+1,t+1}$. □

3.3 Complexity

The number of keys needed in a $2-(v, t+1, 1)$ design can easily be computed. The parameters 2 and 1 in this design are more generally denoted as t and λ respectively. To avoid confusion the parameter t will not be used in this context and always represent the number of colluders. Instead of n the letter b is commonly used and instead of $t+1$ the letter k is used. This means the design used is a 2-design with $b = n$, $k = t+1$ and $\lambda = 1$. Basic block design theory states that

$$bk = vr, \tag{3.6}$$

and

$$\lambda(v-1) = r(k-1). \tag{3.7}$$

Here v is the total number of keys and r is the number of nodes that possesses a particular key, which is independent of which key is chosen. These equations can be rewritten with $\lambda = 1$ as

$$v(v-1) = nt(t+1), \tag{3.8}$$

$$v \approx t\sqrt{n}, \tag{3.9}$$

where $b = n$ and $k = t+1$. This means that a block design uses a total of $O(t\sqrt{n})$ keys. This is an improvement to using Harary graphs, which use $O(t \cdot n)$. A disadvantage of using block designs is that the number of communication routes is $O(t^2)$, while it is $O(t)$ for Harary graphs. Table 3.1 lists all values of $n < 80$ for which a $2-(n, t+1, 1)$ design exists. It lists the number of keys c_b used in such a block design and it also lists the number of keys c_H that is needed when a Harary graph is used. The last line shows the maximum number of colluders t that still allows for safe communication.

n	7	12	26	35	57	70	13	20	50	63	21	30	31	56	57	72	73
c_b	7	9	13	15	19	21	13	16	25	28	21	25	31	49	57	64	73
c_H	11	18	39	53	86	105	26	40	100	126	54	75	93	196	228	288	330
t	2	2	2	2	2	2	3	3	3	3	4	4	5	6	7	7	8

Table 3.1: Values of $n < 80$ for which a key assignment that uses a $2-(n, t+1, 1)$ block design exists for $t > 1$. n is the number of nodes, c_b is the total number of keys in the block design, c_H is the total number of keys in a Harary graph, and t is the maximum number of colluders that still allows for safe communication.

Block designs cannot approximate the lower bound found in Equation (3.4), which is $O(t \cdot \log(n))$.

Chapter 4

Constructions for general n

This chapter addresses the problem of finding efficient constructions for general n. To obtain an efficient construction, existing key assignments will be combined to form a larger construction. Section 4.1 shows how existing constructions may be combined to form a larger construction. It also proves that the combined construction is still t-safe. Section 4.2 estimates the total number of keys, c, needed in such a combined construction when $t = 1$. Section 4.3 estimates the total number of keys, c, needed in such a combined construction when $t > 1$.

4.1 Combining constructions

Suppose the constructions $\mathcal{B}_1$ and $\mathcal{B}_2$ are both t-safe key assignments. Then they can be combined using $t + 1$ extra keys. For example, in Figure 4.1 the blocks $\mathcal{B}_1$ and $\mathcal{B}_2$ are both 2-safe. $\mathcal{B}_1$ has seven nodes. It also has seven keys that are labelled 1 to 7. $\mathcal{B}_2$ is the same construction as $\mathcal{B}_1$. Here the seven keys are labelled 8 to 14. The constructions $\mathcal{B}_1$ and $\mathcal{B}_2$ are connected using $t + 1 = 3$ new keys, labelled 15, 16, and 17. These new keys are labelled in italic.
As an example, it is shown how N_1 and N_2 can safely communicate. After this example it will be shown that such a construction is t-safe in general.
Suppose that N_1 wants to send a message M to N_2. First he splits up the message M into 3 shares: $M = M_A \bigoplus M_B \bigoplus M_C$. It is shown how the share M_A is safely transmitted to N_2. The other shares are transmitted in a similar way.
M_A is sent from N_1 to A_1 using the standard 2-safe protocol in $\mathcal{B}_1$. This means that M_A is split up into 5 shares: $M_A = M_{A_1} \bigoplus M_{A_{5,2}} \bigoplus M_{A_{5,3}} \bigoplus M_{A_{6,2}} \bigoplus M_{A_{6,3}}$. M_{A_1} is sent to A_1 directly with the common key 1. $M_{A_{5,2}}$ is sent to A_1 through the node labelled with (2,5,7), using keys 5 and 2. The other three shares are sent in a similar way. A_1 can then recover M_A, while an adversary of size at most 2 cannot recover M_A. Next, M_A is sent from A_1 to A_2 using the new key 15. Finally M_A is sent from A_2 to N_2 using the 2-safe protocol in $\mathcal{B}_2$. This means that M_A is again split up into 5 shares, so that the transmission of M_A from A_2 to N_2 is 2-safe. Now the only nodes that know M_A are N_1, A_1, A_2, and N_2. To ensure that the communication from N_1 to N_2 is 2-safe, the share M_B is sent through B_1 and B_2, and the share M_C is sent through C_1 and C_2. It will now be proved that such a construction is t-safe, even when multiple blocks are connected.

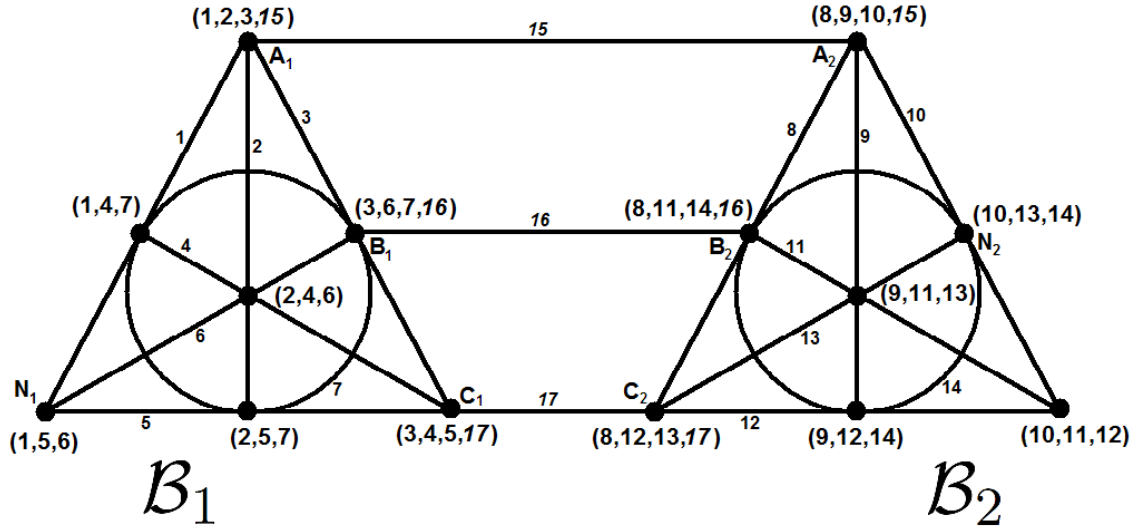


Figure 4.1: The constructions $\mathcal{B}_1$ and $\mathcal{B}_2$ are both 2-safe. They are connected with $t+1=3$ new keys that are labelled in italic. This results in a bigger construction that is still 2-safe as is proved in the text.

Construction 4.1.1. *Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m$ all be t-safe key assignments that use different key sets, with $|\mathcal{B}_i| \geq t+1 \; \forall i$. Pick $t+1$ nodes in each block, labelled $N_{1,1}, N_{1,2}, \ldots, N_{m,t}, N_{m,t+1}$. Assign a new key $k_{new,j}$ to the j-th node of each block. So $\forall i \in \{1, \ldots, m\} \; \forall j \in \{1, \ldots, t+1\}$ the node $N_{i,j}$ gets key $k_{new,j}$.*

Theorem 4.1.2. *The resulting key assignment from Construction 4.1.1 is t-safe.*

Proof. Suppose node A wants to send a message M to node B.

Case 1: A and B are in the same block $\mathcal{B}_i$
A uses the standard protocol in $\mathcal{B}_i$ to send the message M to B. This protocol uses no new keys. That means that none of the nodes in the other blocks, $\mathcal{B}_{l \neq i}$, knows any of the keys used in this protocol. Because $\mathcal{B}_i$ is t-safe this communication is also t-safe.

Case 2: A is in block $\mathcal{B}_i$ and B is in block $\mathcal{B}_l$ where $i \neq l$
In this case the message M is split up into $t+1$ shares: $M = M_1 \bigoplus M_2 \bigoplus \cdots \bigoplus M_{t+1}$. The share M_1 is sent from A to $N_{i,1}$, using the t-safe protocol in $\mathcal{B}_i$. Only old keys from $\mathcal{B}_i$ are used so this transmission is indeed t-safe. The shares $M_2, M_3, \ldots, M_{t+1}$ are sent in the same way to $N_{i,2}, N_{i,3}, \ldots, N_{i,t+1}$ respectively. If A is one of the nodes $N_{i,1}, N_{i,2}, \ldots, N_{i,t+1}$, no transmission is necessary for the corresponding share.
Next, the shares $M_1, M_2, \ldots, M_{t+1}$ are sent from $N_{i,1}, N_{i,2}, \ldots, N_{i,t+1}$ in $\mathcal{B}_i$ to $N_{l,1}, N_{l,2}, \ldots, N_{l,t+1}$ in $\mathcal{B}_l$ respectively, using the keys $k_{new,1}, k_{new,2}, \ldots, k_{new,t+1}$. Finally, the shares $M_1, M_2, \ldots, M_{t+1}$ are sent from $N_{l,1}, N_{l,2}, \ldots, N_{l,t+1}$ to B, using the t-safe protocol in $\mathcal{B}_l$. These last transmissions are again t-safe, because only the old keys in $\mathcal{B}_l$ are used for this.
It remains to be shown that the transmission of $M_1, M_2, \ldots, M_{t+1}$ from $\mathcal{B}_i$ to $\mathcal{B}_l$ is t-safe.

First, note that any partial information on $M_1, M_2, \ldots, M_{t+1}$ determined from the tranmission within $\mathcal{B}_i$ and $\mathcal{B}_l$ is useless. This is because every time a message is split up, all shares but one are determined at random. In particular, the splittings of $M_1, M_2, \ldots, M_{t+1}$ in $\mathcal{B}_i$ are not reused in $\mathcal{B}_l$.
Now consider the transmission of $M_1, M_2, \ldots, M_{t+1}$ from $\mathcal{B}_i$ to $\mathcal{B}_j$. Each of the shares M_j is encrypted by a different key, $k_{new,j}$. To determine M all these shares must be known, which means that an adversary needs all new keys, $k_{new,1}, k_{new,2}, \ldots k_{new,t+1}$, for this. From the way these new keys were assigned, it is clear that no node possesses more than one new key. This means that an adversary of size t or less cannot determine M. So the key assignment obtained from this construction is indeed t-safe. □

In Theorem 4.1.2 all key assignments that are combined need to have size at least $t+1$. Especially for small desired values of n, this constraint is really a restriction on the possible values for n. Therefore another way of extending key assignments is introduced. This is adding one node to an existing t-safe key assignment.

Theorem 4.1.3. *Let $\mathcal{B}$ be a t-safe key assignment on $n \geq t+1$ nodes. Then this key assignment can be extended to a t-safe key assignment on $n+1$ nodes with $t+1$ extra keys.*

Proof. Let $N_1, N_2, \ldots, N_{t+1}$ be nodes in the t-safe key assignment $\mathcal{B}$, and let N_{n+1} be the new node to be added. Assign a new key $k_{new,j}$ to each of the nodes $N_1, N_2, \ldots, N_{t+1}$, so $\forall j \in \{1, \ldots, t+1\}$ N_j gets $k_{new,j}$. Also, assign all of these new keys to the new node N_{n+1}. Then the obtained key assignment can be proved t-safe.
For communication between two old nodes the standard t-safe protocol in $\mathcal{B}$ is used. This protocol is still t-safe, because it uses no new keys, and N_{n+1} possesses no old keys.
For communication between the new node N_{n+1} and another node N_i, the following protocol is used. N_{n+1} splits up the message M into $t+1$ shares, $M = M_1 \bigoplus M_2 \bigoplus \cdots \bigoplus M_{t+1}$. Each share, M_j, is sent to a different node N_j, encrypted by the new key $k_{new,j}$. Next, each of the M_j are split up and transmitted from N_j to N_i, using the t-safe protocol in $\mathcal{B}$. If N_i is the same node as N_j for some j, the transmission of the corresponding share is not necessary. Because the transmission of the shares M_j within $\mathcal{B}$ is t-safe, the only nodes that can determine M_j are N_{n+1}, N_j, and N_i. The nodes N_{n+1} and N_i are not in the adversary because they are the sender and receiver of M. An adversary needs all shares M_j to determine M. This means that it must consist of at least $t+1$ nodes. So the new key assignment is indeed t-safe. If the message M is sent from N_i to N_{n+1} instead, the protocol is used in reverse. □

4.2 Combining blocks for $t = 1$

With the constructions from Section 4.1 efficient 1-safe key assignments can be made for any number of nodes n. This section estimates the total number of keys, c, needed for such a key assignment.
The available blocks from Section 2.2 have size $n = \binom{c}{\lfloor \frac{c}{2} \rfloor}$. The first couple of values for n are listed in Table 2.2. It is notable that n doubles or almost doubles when c is increased by 1. The exact behaviour of n when c increases by 1 can be derived. n_c is the number of nodes that corresponds to a block assignment with c keys:

$$\frac{n_{c+1}}{n_c} = \frac{\binom{c+1}{\lfloor \frac{c+1}{2} \rfloor}}{\binom{c}{\lfloor \frac{c}{2} \rfloor}} = \frac{(c+1)!}{(\lfloor \frac{c+1}{2} \rfloor)!(\lceil \frac{c+1}{2} \rceil)!} \frac{(\lfloor \frac{c}{2} \rfloor)!(\lceil \frac{c}{2} \rceil)!}{c!} = \frac{c+1}{\lceil \frac{c+1}{2} \rceil} = \begin{cases} 2 & c \text{ is odd} \\ 2\frac{c+1}{c+2} & c \text{ is even} \end{cases}.$$

So $\frac{n_{c+1}}{n_c} \leq 2$ and tends to 2 as c grows. This gives an obvious way of combining blocks, which is not optimal, but is good for most n. It resembles the binary representation of n. A network for n nodes is constructed as follows. First the largest c_1 is determined such that $\binom{c_1}{\lfloor \frac{c_1}{2} \rfloor} \leq n$. This block with c_1 keys is used for the first $\binom{c_1}{\lfloor \frac{c_1}{2} \rfloor}$ nodes. This process is repeated for the remaining $n - c_1$ nodes. This gives a block with $c_2 < c_1$ keys, which is combined with the first block. Since every smaller block is at least half the size of the previous block this process is repeated at most c_1 times. Taking $c_1 \approx \log_2 n$ from Equation (2.3), the total number of keys c becomes at most:

$$\begin{aligned} c &\approx 2 + \log_2 n + \log_2 \frac{n}{2} + \log_2 \frac{n}{4} + \cdots = 2 + (\log_2 n) + (\log_2 n - 1) + \ldots \\ &\approx 2 + (\log_2 n)^2 - \frac{\log_2 n}{2}(\log_2 n + 1) = 2 + \frac{(\log_2 n)^2 - \log_2 n}{2} \approx \frac{(\log_2 n)^2}{2}, \end{aligned} \tag{4.1}$$

where the number of terms is estimated to be $\log_2 n$, and the "2+" is due to the two extra keys required for combining. This means that a general construction for $t = 1$ exists that uses $O((\log_2 n)^2)$ keys.

Table 4.1 lists the total number of keys, c, needed for a 1-safe key assignment on n nodes, using the most efficient combinations of the designs presented in Section 2.2. The combining is done as explained in Section 4.1.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
c	2	3	5	7	4	6	8	9	5	7	9	10	11	12	11	13	13	14	6

Table 4.1: The number of keys required for key assignments with $t = 1$ and $n \leq 20$. c is the number of keys required when a combination of key assignments is used.

All combinations presented in Table 4.1 are made in the way proposed in this section. However, this is not always optimal. For example, when $n = 140$, it is more efficient to combine two key assignments of size 70 than to combine key assignments of size 126, 10, 2, and 2. The first construction uses a total of 18 keys, while the second uses 20 keys. These numbers follow from Table 2.2 and Theorem 4.1.2. Figure 4.2 in Section 4.3 shows the number of keys required for all $n \leq 70$.

In [4] a slightly different construction for $t = 1$ is discussed in Chapter 3. This construction uses one key more than the constructions presented in Chapter 2 for the values of n listed in Table 2.2. For other values of n it uses fewer keys than the constructions presented in this section. In general, it uses $O(\log_2 n)$ keys, which is better than the $O((\log_2 n)^2)$ keys achieved in this section.

4.3 Combining blocks for $t > 1$

With the constructions from Section 4.1 t-safe key assignments can be made for any number of nodes n. These will still use fewer keys than key assignments derived from Harary graphs. This section estimates the total number of keys, c, needed fo such a key assignment. Suppose two block designs $\mathcal{B}_1$ and $\mathcal{B}_2$ of size n_1 and n_2 exist that are both t-safe. This means these designs use at most $(t+1)\sqrt{n_1}$ and $(t+1)\sqrt{n_2}$ keys respectively, as can be derived from Equation 3.8; the quadratic formula gives the following positive solution for the total number of keys $c = v$:

$$c = \frac{1 + \sqrt{1 + 4nt(t+1)}}{2}, \tag{4.2}$$

which can be estimated by

$$\begin{aligned} c &= \frac{1 + \sqrt{1 + 4nt(t+1)}}{2} \leq \frac{1 + \sqrt{1 + 4nt(t+1) + 4\left(n - \sqrt{n}\right)(t+1)}}{2} \\ &= \frac{1 + \sqrt{\left(2\sqrt{n}(t+1) - 1\right)^2}}{2} = (t+1)\sqrt{n}. \end{aligned} \tag{4.3}$$

Equation (4.3) gives an actual upper bound for c, whereas Equation (3.9) is a more accurate estimate for c. The upper bound from Equation (4.3) will be used to derive upper bound on the number of keys needed for the constructions presented in this section.

The total number of keys needed for m connected networks is $O(t\sqrt{m \cdot n})$, where in the worst case the networks are of similar size. The number of communication routes now becomes $O(t^3)$.

An open question remains how many block designs must be combined to obtain a network with exactly n nodes. To make an estimate, suppose that all designs that satisfy Equation (3.5) exist. Then $v = \begin{cases} 1 + kt(t+1) \\ t + 1 + kt(t+1) \end{cases} k \in \mathbb{N}$.

When Equation(3.8) is solved for n and the given values of v are used this gives the following possibilities for the number of nodes n:

$$n = \begin{cases} t(t+1)k^2 + k \\ t(t+1)k^2 + (2t+1)k + 1 \end{cases} k \in \mathbb{N}. \tag{4.4}$$

As was shown in Theorem 4.1.3 a block of size 1 may be added at the cost of $t+1$ extra keys. One more possibility is added, namely the complete graph on $t+1$ points, which is t-safe. The two sequences in Equation (4.4) grow quadratically. Therefore, finding the number of blocks that must be combined to obtain a communication graph on n points is related to Waring's problem [5]. The sequences in Equation (4.4) grow at a rate equal to the $(2(t^2+t+1))$-gonal numbers, if only the quadratic terms are considered. By Fermat's Polygonal Number Theorem every natural number can be represented as the sum of at most p p-gonal numbers [6]. It is conjectured here that a t-safe key assignment can be constructed using at most $2(t^2+t+1)$ block designs. Note that this conjecture is stronger than assuming that every number can be written as the sum of at most $2(t^2+t+1)$ numbers from the sequences in Equation (4.4), after the number 1 has been added to that sequence. This is because Equation (3.5) is only

a sufficient condition for the existence of a block design for $t = 2$, 3, and 4, so for $t > 5$ block designs do not exist for all n that satisfy Equation (4.4).

With this conjecture the maximum number of keys needed, is at most the number of keys necessary to combine $2(t^2 + t + 1)$ equal size blocks, since

$$\max_{n_1,\ldots,n_m} \{\sqrt{n_1} + \sqrt{n_2} + \cdots + \sqrt{n_m} \quad | \quad n_1 + n_2 + \cdots + n_m = n\} = \sqrt{m \cdot n}. \tag{4.5}$$

The total number of keys, c, is then at most

$$c \leq (t+1)\left(1 + \sqrt{2n(t^2+t+1)}\right) \leq (t+1)^2\sqrt{2n}. \tag{4.6}$$

where Equation (4.3) is used to estimate the maximum number of keys in one block.

Tables 4.2, 4.3, and 4.4 give an indication of the efficiency of combining block designs. They list the total number of keys, c_b, needed for a t-safe key assignment on n nodes, using the most efficient combinations of block designs. They also list the number of keys, k_H, that are needed in a Harary graph.

Each table starts with the value $n = t+1$. In practice a construction for this value does not make much sense, since it implies that all nodes but one are corrupted. In that case at least one of the communicating nodes is corrupted. However, when block designs are combined these blocks are useful building elements. For example, in Table 4.2 the key assignment for $n = 16$ is obtained as follows: the block design for $n = 12$ is combined with the block for $n = 3$. This results in a key assignment for $n = 15$ that uses $9 + 3 + 3 = 15$ keys; 9 for the first block, 3 for the second block, and 3 for combining. Finally, one additional node is added using 3 new keys to connect it to the existing network. This requires a total of 18 keys.

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
c_b	3	6	9	9	7	10	12	13	16	9	12	15	15	18	21	18	19	22
c_H	-	6	8	9	11	12	14	15	17	18	20	21	23	24	26	27	29	30

Table 4.2: The number of keys required for key assignments with $t = 2$ and $n \leq 20$. c_b is the number of keys required when a combination of block designs is used; c_H is the number of keys required when a Harary graph is used.

n	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
c_b	6	10	14	18	16	20	24	28	22	13	17	21	25	23	27	31	16
c_H	-	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40

Table 4.3: The number of keys required for key assignments with $t = 3$ and $n \leq 20$. c_b is the number of keys required when a combination of block designs is used; c_H is the number of keys required when a Harary graph is used.

It is remarkable that the construction for $t = 3$ and $n = 20$ in Table 4.3 requires fewer keys than the construction for $t = 2$ and $n = 20$ in Table 4.2. This means that in constructing 2-safe key assignments it is sometimes more efficient to use 3-safe blocks. Implementing this improvement gives the graph in Figure 4.2. It gives the total number of keys required for Harary graphs and for combined block designs. The graph for $t = 1$ is also included, but that graph uses constructions that are asymptotically logarithmic in n. The other constructions are asymptotically proportional to $\sqrt{n}$.

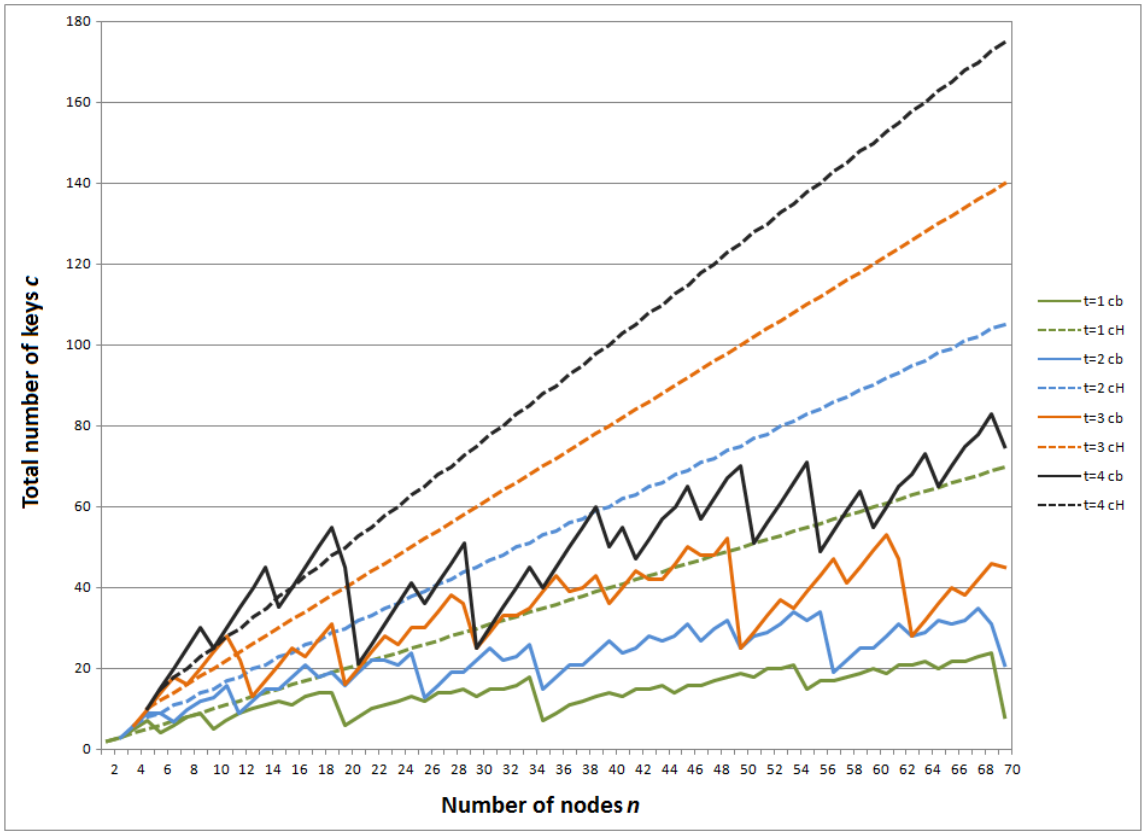


Figure 4.2: This figure shows the total number of keys, c, needed for a t-safe key assignment on n nodes. Graphs are plotted for t = 1,2,3, and 4. The dashed lines give the number of keys needed when a Harary graph is used. The solid lines give the number of keys needed when a combination of block designs is used.

n	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
c_b	10	15	20	25	30	25	30	35	40	45	35	40	45	50	55	45
c_H	-	15	18	20	23	25	28	30	33	35	38	40	43	45	48	50

Table 4.4: The number of keys required for key assignments with $t = 4$ and $n \leq 20$. c_b is the number of keys required when a combination of block designs is used; c_H is the number of keys required when a Harary graph is used.

4.4 t proportional to n

In a practical situation one would expect the number of colluders t to be proportional to the number of nodes n. For example, in a computer network the corrupted nodes are usually the computers that are infected by the same virus. If t is proportional to n the estimate in Equation (4.6) is no longer valuable, because it is asymptotically worse than a construction that uses Harary graphs, which uses $O(t \cdot n)$ keys. The best block designs are still more efficient than Harary graphs by Equation (4.3), but the combining method may not be efficient for all values of n anymore.

Chapter 5

Conclusion and discussion

This report presents constructions for secure communication in a network with t colluders. For $t = 1$ these constructions use all $\lfloor \frac{c}{2} \rfloor$ size subsets of a set of size c. To obtain a general construction, multiple constructions are combined to form larger constructions. For $t > 1$ the constructions use $2 - (v, t + 1, 1)$ block designs or a combination of multiple such block designs.

The presented constructions are an improvement on existing constructions for the total number of keys needed. The best constructions require a total number of $O(\log_2 n)$ keys for n nodes when the number of colluders t equals 1. These constructions are proved to be optimal. When the number of colluders t is larger than 1, the best constructions require a total number of $O(t \cdot \sqrt{n})$ keys for n nodes. This may not be optimal. The general case is conjectured to require no more than $(t + 1)^2 \cdot \sqrt{2n}$ keys.

Improvements may be made for $t > 1$; either on the lower bound in Equation (3.4) or by finding constructions that approach this lower bound. A computer simulation may find a key assignment that uses fewer keys than a block design.

Another possible improvement would be constructions that require a similar number of keys as combined block designs, but use fewer communication routes than $O(t^3)$.

Bibliography

[1] Yvo Desmedt, Henk van Tilborg, and Huaxiong Wang, "Key Distribution Schemes and Harary Graphs", private communication

[2] Emanuel Sperner, "Ein Satz ber Untermengen einer endlichen Menge" , Mathematische Zeitschrift 27, pages 544548, 1928

[3] Handbook of Combinatorial Designs, author, edition, pages, ...

[4] Yvo Desmedt, Niels Duif, Henk van Tilborg, and Huaxiong Wang, "Bounds and constructions for key distribution schemes", Advances in Mathematics of Communications, Volume 3, Number 3, August 2009

[5] Eric W. Weisstein, "Waring's Problem." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/WaringsProblem.html`

[6] Eric W. Weisstein, "Fermat's Polygonal Number Theorem." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/FermatsPolygonalNumberTheorem.html`

[7] J.H. van Lint, "Introduction to coding theory", Graduate texts in Mathematics 86, Springer Verlag, Berlin, 1982