Distributing the Encryption and Decryption of a Block Cipher

Keith M. Martin[1], Rei Safavi-Naini[2], Huaxiong Wang[3] and Peter R. Wild[1]

[1]Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, U.K.

[2]School of Information Technology and Computer Science
University of Wollongong, Northfields Avenue, Wollongong 2522, Australia

[3] Department of Computing
Macquarie University, North Ryde, NSW 2109, Australia

Abstract

In threshold cryptography the goal is to distribute the computation of basic cryptographic primitives across a number of nodes in order to relax trust assumptions on individual nodes, as well as to introduce a level of fault-tolerance against node compromise. Most threshold cryptography has previously looked at the distribution of public key primitives, particularly threshold signatures and threshold decryption mechanisms. In this paper we look at the application of threshold cryptography to symmetric primitives, and in particular the encryption or decryption of a symmetric key block cipher. We comment on some previous work in this area and then propose a model for shared encryption / decryption of a block cipher. We will present several approaches to enable such systems and will compare them.

1 Introduction

In an increasingly networked and distributed communications environment there are more and more practical situations where the ability to distribute a computation between a number of different network nodes is required. There are several reasons why such a distribution is desirable. These include reasons of efficiency (separate nodes perform distinct tasks), fault-tolerance (if some nodes are unavailable then others can perform the task) and security (the trust required to perform the task is shared between nodes).

The field of *threshold cryptography* [12] is concerned with distributing the ability to compute cryptographic primitives primarily for the latter two reasons. Although some work has been done on sharing general functions [11], particular attention has been paid to investigating ways of sharing the computation of specific public key cryptographic primitives. Several papers, such as [4, 10], have looked at the shared generation of public key pairs. Most of the work in this area, for example [5, 18], has investigated distributing the ability to compute digital signatures. Others, such as [8, 19], have also considered sharing public key decryption. As well as being interesting in their own right, distributed public key primitives have applications to many other areas such as electronic voting [17]. One of the reasons why public key cryptography has almost exclusively been investigated in this regard is that the inherent algebraic structure, and resulting homomorphic properties, behind some public key algorithms (particularly discrete-logarithm-based algorithms such as ElGamal) lends itself naturally to distributed computation. In such algorithms it can be relatively straight forward to perform partial component operations and then combine them algebraically to obtain a combined result.

Symmetric key cryptography, on the other hand, is generally based on algorithms that rely on a *lack* of algebraic structure in order to guarantee their security. This is why many symmetric key cryptographic primitives such as block ciphers, stream ciphers and message authentication codes have not been subject to much study within the context of threshold cryptography. In [7] the authors discussed a technique for sharing the computation of a block cipher using a new technique that they referred to as *sequence sharing*.

In this paper we continue the investigation of how to share the computation of a block cipher. We will revisit sequence sharing, note that this technique has already been investigated under a different name, and demonstrate that existing results generalise and unify the results proven in [7]. We will further propose a method for implementing this technique more efficiently. We then consider a number of other ways in which the computation of a block cipher could be distributed amongst several network entities and compare them. Finally we propose a number of natural extensions of the block cipher sharing schemes proposed in this work.

2 Sharing Block Ciphers

Informally when we refer to *sharing a block cipher* we mean that either the encryption or the decryption of a message sent using that block cipher is a process to be distributed amongst a group of entities. In *shared encryption* a group of senders co-operatively compute the ciphertext, which they then send to a single receiver, who decrypts it. In *shared decryption* a single sender encrypts a message using the block cipher and sends it to a group of receivers, who must co-operate in order to decrypt the ciphertext. We do not specify the exact nature of the co-operation and consider a number of different methods in this paper. In each case we associate an access structure with the distributed end of the process. We refer to the subsets of senders or receivers that belong to the access structure as authorised subsets. In most cases it is also possible to extend the methods in this paper to distribute the computation at both ends of the communication channel.

It is worth noting that shared encryption and shared decryption differ in the nature of the communication link. In shared encryption, while a ciphertext may have originated from a variety of subgroups it involves communication from one subgroup to the receiver. The receiver must succeed in decrypting the ciphertext if it has been produced by an authorised subset and detect a forgery otherwise.

In shared decryption a ciphertext may be received by a variety of subgroups. Such a group of receivers can decrypt to the corresponding message if and only if they are an authorised subset. There may be several subgroups that can decrypt the ciphertext so this involves communication from the sender to many subgroups.

To illustrate the idea, consider a possible block cipher sharing scenario. The directors of a company have set up a system to use block cipher sharing to communicate with their chairman, who is at a remote location. The goal of shared encryption is for any three (say) directors to be able jointly to send an encrypted message to the chairman, and that by decrypting it using the correct process, the chairman can be sure that some group of (at least) three directors (not one, not two) must have co-operated in the encryption. For shared decryption, the goal is for the chairman to be able to send a message that requires any three (at least) directors to co-operate before it can by decrypted. One or two directors will not suffice.

2.1 Motivation

Brickell *et al* [7] first proposed the sharing of block ciphers, motivated by a desire to extend the domain of cryptographic primitives to which threshold cryptography can be applied. We briefly consider here a more fundamental question - namely, *exactly what purpose does sharing a block cipher computation serve?*

If we assume that the underlying block cipher is ”secure” (in whatever sense is appropriate) then clearly a simple application of the block cipher suffices to provide message confidentiality. There are two main benefits of sharing a block cipher computation.

The first is a direct security benefit. By requiring a group to co-operate in the shared encryption (decryption) we have distributed the trust required to perform this computation. This reduces the inherent dangers of compromise of one of the participants.

The second benefit is the increased availability of the distributed end of the communication in the presence of faults. In a conventional point to point communication channel, if the sending node is unavailable then the communication cannot take place. By distributing the ability to send a message, the system can cope with some nodes being unavailable by using others to send the message. Of course, this benefit could be realised by having multiple nodes that each have the capability to send the entire message. However by increasing the number of nodes with this capability, we would be increasing the threat of compromise to the system by introducing multiple “single points of failure”. This increased threat is avoided in the techniques we describe here.

2.2 Options for shared encryption

We note a couple of methods that could be used to meet similar goals to shared encryption.

Using the previous example of the chairman and directors, one option might be for any one director to send a conventionally encrypted message to the chairman, and for three directors to then use a group signature scheme [9] to sign the message ”we, the following three directors approve the sending of the encrypted message”. Although this would not directly meet the particular goal of shared encryption, it could be argued that if the purpose of shared encryption is to provide a degree of assurance to the receiver that the senders have jointly agreed that the ciphertext should be sent to the receiver, then this might suffice. However, a significant problem with this solution is that a suitable infrastructure must be in place to manage the signing and verifying of group signatures. The solutions we describe here use symmetric primitives alone.

A simple solution that appears to directly meet the goals of shared encryption is as follows. Each of the senders shares a symmetric key with the receiver. To send a message to the receiver, each of the senders encrypts the message using the key that they share with the receiver and sends this ciphertext directly to the receiver. The receiver accepts a “shared encryption” only if they receive ciphertexts from each member of an authorised set of senders. The problem with this “solution” is that it is not an option for shared decryption and there is increased bandwidth required to send multiple ciphertexts to the receiver. However, we note that the third technique that we describe here (see Section 6) also suffers from the latter problem.

2.3 Trust assumptions

We assume the existence of either a single sender and a group of receivers, or a group of senders and a single receiver. We also assume that some initial key material is securely distributed to the participants by a trusted dealer (or key distribution centre) but that the participants do not have access to these secure channels to the dealer after this initialisation process has taken place. As in any symmetric cryptosystem, we rely on participants keeping their own key material secret. If participants reveal their key material or it is compromised then this may change the composition of the access structure. We do not consider here how to counteract this and so consider any group to be authorised if they have the necessary key material no matter how it has been obtained.

A conceptually simple solution to providing shared encryption (decryption) involves applying simple secret sharing techniques (see next section) to the key. For example, for shared encryption each sender could be given a share of the encryption key. To send a message they could forward the message plus their shares of the encryption key to a trusted combiner, who reconstructs the encryption key, encrypts the message and sends it to the receiver. However, the problems with this solution are that such a powerful trusted combiner must be created, maintained, and be available online each time a message is sent, and that each of the participants at the distributed end of the communication would need to have their own secure channel to this trusted combiner. We do not assume the existence of a trusted combiner of key material.

We assume that when an authorised group wish to encrypt a message or decrypt a some ciphertext they cooperate by taking part in a protocol. This protocol enables them to perform the distributed computation of the cipher. We do not assume that they necessarily have secure channels providing confidentiality between them; that is, they may communicate via broadcast channels when executing the protocol. Thus, when it comes to performing the distributed computation, the information that the participants exchange with one another may be public.

When used for shared decryption the third scheme we describe (see section 6) may reveal information about the message to an adversary if more than one authorised group decrypts the ciphertext. This problem is overcome if there exists an entity who has a secure channel providing confidentiality with each of the participants, and who is trusted to act as a local combiner of shares of the message. This local combiner is much weaker than the dealer of the initialisation material, since they are only active at one end of the communication channel. Further, since this local combiner does not manage key material (as in the case just discussed), there may be more than one such entity and it is possible that they could be one of the receivers.

The single sender (or receiver) is trusted to follow their part in the protocol correctly since this is merely encrypting a message or decrypting ciphertext using their key. At the distributed end of the communication, groups of entities may or may not trust one another. It is the role of the access structure to ensure that any authorised group that takes action to encrypt or decrypt a message includes sufficiently many trustworthy and uncompromised participants. Ideally a misbehaving or compromised entity acting within a group in the access structure should be no more capable of fraudulent shared encryption or decryption than a group not in the access structure. Specifically, such an entity should not be able to manipulate shared encryption by an authorised group so that a ciphertext of an unintended message is produced, nor manipulate shared decryption by an authorised group to obtain information about the message that the other members of the group cannot obtain. Whereas it might be desirable that false computation by any one (or more) of the distributed entities should be detected by the others, not all our schemes have this property. We do not consider the task of addressing this by adding robustness to these protocols (such as has been done for several threshold public key primitives [19]) in this paper, but note that this topic might be worthy of further investigation.

2.4 Security

We consider sharing a block cipher to provide confidentiality. A similar analysis may be made for sharing a block cipher to provide authentication. As with any block cipher designed to provide confidentiality, shared encryption and shared decryption have the security goal that an adversary should not be able to deduce significant information about the message corresponding to an intercepted ciphertext. Whereas in the standard use of a block cipher an adversary may have access to corresponding message and ciphertext pairs under the key agreed upon by the sender and receiver, in block cipher sharing two situations may arise. If the participants in the shared computation have secure channels of communication between them they may relay the results of partial computations amongst themselves without these values being revealed to the adversary. On the other hand, communication between the participants may use public channels and an adversary may in addition have access to the intermediate results of partial computations made by the participants of an authorised group (using their respective shares of the key material). With this proviso, the security of block cipher sharing may be measured by any method appropriate to block ciphers.

We assume that there is a probability distribution on the messages, representing their statistical frequency in communications between the sender and receiver, and that this is known to the adversary. An adversary may use this probability distribution to predict the contents of a message without having observed a transmitted ciphertext. We also assume that the key material shared by the participants of a block sharing scheme is secret so that an adversary only has probabilistic information about the key. These two probability distributions together determine a probability distribution on the ciphertexts.

We assume the adversary has limited computational resources and has access to some intercepted ciphertext and a limited amount of message and ciphertext pairs that correspond under the key. An adversary may use these resources to revise his estimates of the likelihood of any particular message corresponding to an intercepted ciphertext obtained using that key and his probabilistic information about the shared key. An adversary may include an unauthorised subset of (compromised) participants who have information about the key. In this context, we consider to be authorised any subset of participants who can use their knowledge about the key, and the intermediate results of partial computations of some members of an authorised subset, in order to complete the computation. We consider a block cipher sharing scheme to be secure if the existence of a successful adversary implies the existence of an adversary of comparable power successful against a block cipher that is considered secure (with respect to an appropriate measure). Thus we measure the security of a block cipher sharing scheme by a reduction to the security of an appropriate block cipher.

3 Definitions

In this section we give a formal definition of what we mean by sharing block ciphers. We begin by defining their components: block ciphers and secret sharing schemes.

3.1 Block Ciphers

A *block cipher* is a triple of sets K, M, C and a mapping $F : K \times M \to C$ such that for all $k \in K$ the mapping $F_k : M \to C$ given by $F_k(m) = F(k, m)$ for all $m \in M$ is reversible. The key k also determines the reverse mapping $F_k^{-1} : C' \to M$, where $C' = F_k(M) = \{F_k(m) : m \in M\}$, such that $F_k^{-1}(c) = m$ if and only if $c = F_k(m)$ for all $m \in M$ and all $c \in C'$. When each mapping F_k is a permutation these inverse mappings constitute a corresponding (inverse) block cipher $F^{-1} : K \times C' \to M$.

The elements of K are the keys and the elements of M are the messages. The ciphertext $c = F_k(m) \in C$ is the encryption of $m \in M$ under key $k \in K$. The key k is agreed in secret by the sender and receiver and it is used to transform the message into the ciphertext to be transmitted on the public channel. Thus, if the function F has appropriate properties, the secret key may be used to protect the confidentiality of the message.

We assume that an adversary who intercepts a ciphertext has only probabilistic information about the message and the key. That is, there are probability distributions, known to the adversary, on M and K. These determine a probability distribution on C. It is desirable that, without knowlegdge of the key, knowledge of the ciphertext does not enable the prediction of the message and that the secrecy of the key is not compromised by observation of the ciphertexts or knowledge of corresponding message and ciphertext pairs. We consider a block cipher to be secure if there is no efficient algorithm that uses (i) knowledge of the cipher system; (ii) knowledge of the probability distribution on the key space; (iii) knowledge of the message statistics; and (iv) limited access to corresponding message and ciphertext pairs for the same key; to determine useful information about the message corresponding to any further ciphertext.

3.2 Secret Sharing

We are interested in communications environments where there are several entities that wish to cooperate in groups to send and receive communications securely.

Let $\mathcal{P}$ be a set $\{P_1, \ldots, P_n\}$ of n entities, called *participants*, who take part in sending and receiving communications. An *access structure* on $\mathcal{P}$ is a collection $\mathcal{A}$ of subsets of $\mathcal{P}$. A subset $A \in \mathcal{A}$ is called an *authorised set* (of participants). We only consider *monotone* access structures characterised by the property that if $A \in \mathcal{A}$ and $A \subseteq A'$ then $A' \in \mathcal{A}$. Thus any set of participants that contains an authorised subset is authorised. In a monotone access structure a *minimal authorised subset* is a subset $A \in \mathcal{A}$ such that $A \backslash \{a\} \notin \mathcal{A}$ for all $a \in A$, and a *maximal unauthorised subset* is a subset $A \notin \mathcal{A}$ such that $A \cup \{a\} \in \mathcal{A}$ for all $a \in \mathcal{P} \backslash A$. We denote the set of all minimal authorised subsets of $\mathcal{A}$ by $\mathcal{A}^-$ and the set of all maximal unauthorised subsets of $\mathcal{A}$ by $\mathcal{A}^+$.

For $i = 1, \ldots, n$, let S_i denote a set of elements, called *shares*, corresponding to participant $P_i \in \mathcal{P}$. A *secret sharing scheme* for the key set K on the set of participants $\mathcal{P}$ is a subset D of

$K \times S_1 \times \ldots \times S_n$ together with a probability distribution defined on D. If $(k, s_1, \ldots, s_n) \in D$ than we say that key k is shared among the participants $P_1, \ldots, P_n$ who hold shares $s_1, \ldots, s_n$ respectively. The probability distribution on D induces a probability distribution on K and each of S_i, $i = 1, \ldots, n$. The set D is a secret sharing scheme for K with respect to the access structure $\mathcal{A}$ on $\mathcal{P}$ if

$$H(K|S_{i_1}, \ldots, S_{i_t}) \begin{array}{ll} = 0 & \text{if} \{P_{i_1}, \ldots, P_{i_t}\} \in \mathcal{A} \\ > 0 & \text{if} \{P_{i_1}, \ldots, P_{i_t}\} \notin \mathcal{A} \end{array}$$

for all subsets $\{P_{i_1}, \ldots, P_{i_t}\} \subseteq \mathcal{P}$, where H is Shannon's entropy function. Thus the shares held by a subset of the participants determine the key if and only if the subset is authorised. That is, given $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ and shares $s^{'}_{i_1}, \ldots, s^{'}_{i_t}$ where $s^{'}_{i_j} \in S_{i_j}$ for $j = 1, \ldots, t$ there is a $k' \in K$ such that $k = k'$ for every $(k, s_1, \ldots, s_n) \in D$ with $s_{i_j} = s^{'}_{i_j}$ for $j = 1, \ldots, t$ if and only if $\{P_{i_1}, \ldots, P_{i_t}\} \in \mathcal{A}$. We say that an authorised subset of participants $P_{i_1}, \ldots, P_{i_t}$ pool their shares $s^{'}_{i_1}, \ldots, s^{'}_{i_t}$ to get the key k'.

If $H(K|S_{i_1}, \ldots, S_{i_t}) = H(K)$ whenever $\{P_{i_1}, \ldots, P_{i_t}\} \notin \mathcal{A}$ we say D is *perfect*. In this case any unauthorised subset of participants gains no information about the key by pooling their shares. In perfect secret sharing schemes it can be seen that any share must be at least the size of the secret, in other words $H(S_i) \geq H(K)$ for $i = 1, \ldots, n$. If equality in this bound is met for each share set then the secret sharing scheme is called *ideal* [6].

Secret sharing schemes defined on n participants, whose access structure consists of all sets of size at least t are referred to as (t, n)-*threshold schemes*. The well-known (t, n)-threshold schemes defined by Shamir [26] are examples of ideal threshold schemes.

3.3 Block Cipher Sharing

Block cipher sharing can be thought of as an extension of secret sharing schemes. The set K is the set of keys of a block cipher $F : K \times M \to C$. Now, instead of pooling the shares $s^{'}_{i_1}, \ldots, s^{'}_{i_t}$ to determine the key k' an authorised subset who wants to encrypt a message m uses them as input to a calculation of $F(k', m)$. Any calculation involving $s^{'}_{i_j}$ is performed by P_{i_j} in secret, but the result of such a calculation may be communicated to other participants and it may then be used in subsequent calculations by another participant or some public function. This provides an extra dimension to block cipher sharing. Whereas in secret sharing, for any given sharing of the key, the shares held by any authorised subset of participants determine the same key, in block cipher sharing a different authorised subset might use their shares to encrypt a different message $m^{'} \in M$. In block cipher sharing the key is not made public and the shares may be reused to encrypt several messages. Note, however, that in block cipher sharing some results of intermediate calculations, *i.e.* partial encryptions, may be public. We are interested in the case when knowledge of such partial encryptions does not help an unauthorised set determine the encryption or decryption of a message.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of participants. Let $X, S_1, \ldots, S_n$ be sets. For $\mathbf{s} = (s_1, \ldots, s_n) \in S_1 \times \ldots \times S_n$ and $A = \{P_{i_1}, \ldots, P_{i_t}\} \subseteq \mathcal{P}$, we write $(A, \mathbf{s}_A) = \{(P_{i_1}, s_{i_1}), \ldots, (P_{i_t}, s_{i_t})\}$ and refer to this set as the assignment of shares to the subset A. We use the term *distributed algorithm* with respect to $X, \mathcal{P}, S_1, \ldots, S_n$ to refer to an algorithm $\mathcal{G}$ that takes as input an element $x \in X$ and a subset $A \subseteq \mathcal{P}$ and assignment $(A, \mathbf{s}_A)$ and is performed in a sequence of steps by the subset of participants A such that each step is a calculation by one participant in the subset using their assigned share. Each step may produce an intermediate output that is communicated to (all) other participants and may be used in later steps. Thus the output $\mathcal{G}(x, A, (A, \mathbf{s}))$ is calculated via a sequence $b_1, \ldots, b_r$, where r is the number of steps, b_j, $j = 1, \ldots, r-1$, are the intermediate outputs or partial computations sent to the participants and $b_r = \mathcal{G}(x, A, (A, \mathbf{s}))$ is the final output. For each $j = 1, \ldots, r$, $b_j = f_j(s_{i_j}, x, b_1, \ldots, b_{j-1})$, where s_{i_j} is the share held by the participant P_{i_j} who performs step j and f_j denotes the partial computation at step j.

A *block cipher sharing scheme for encryption* with respect to the block cipher $F : K \times M \to C$ and the access structure $\mathcal{A}$ on $\mathcal{P} = \{P_1, \ldots, P_n\}$ is a secret sharing scheme $D \subseteq K \times S_1 \times \ldots \times S_n$ for K with respect to $\mathcal{A}$ and a distributed algorithm $\mathcal{G}$ with respect to $M, \mathcal{P}, S_1 \times \ldots \times S_n$ such that for all $(k', s'_1, \ldots, s'_n) \in D$ and for all $A = \{P_{i_1}, \ldots, P_{i_t}\} \in \mathcal{A}$ and all $m \in M$, $\mathcal{G}(m, A, (A, \mathbf{s}_A)) = F(k', m)$. Similarly, a *block cipher sharing scheme for decryption* has $\mathcal{G}(c, A, (A, \mathbf{s}_A)) = F_{k'}^{-1}(c)$ whenever $c \in F_{k'}(M)$.

The security of a block cipher sharing scheme for encryption is measured in terms of the ability of an (unauthorised) interceptor to determine m from c. The interceptor may have observed the intermediate values $b_1, \ldots, b_{r-1}$ (and, possibly, may have knowledge of some of the shares $s'_1, \ldots, s'_n$) and may also have observed the partial computations and outcomes of several previous shared encryptions or decryptions. Clearly, the block cipher sharing scheme can be no more secure than the block cipher F itself as observations of the shared computations provide all the information that is obtained by observations of the block cipher alone and an attack that compromises F also compromises the sharing scheme based on F. Ideally, the block cipher sharing scheme should be no less secure than the block cipher itself. We are interested in the case where the sharing of the block cipher, *i.e.* where values b_i may be known or some shares s'_i are known, opens no new avenues of attack.

For each maximal unauthorised set $B \in \Gamma^-$ and participant $P_i \in \mathcal{P} \backslash B$, the participants in B can use the shares that they hold of a key k' to define a mapping $F_{\{s'_j \,:\, P_j \in B\}, Pi} : S_i \times M \to C$ such that $F_{\{s'_j \,:\, P_j \in B\}, P_i}(s'_i, x) = F(k', x)$ for $x \in M$ where s'_i is the share of k' held by P_i. For, in the algorithm given above, the participants have all the required values for the calculation except s'_i. Clearly these mappings are themselves block ciphers. An attack on F can be turned into an attack by the participants of B on $F_{\{s'_j \,:\, P_j \in B\}, P_i}$ since a given distribution of shares determines k' and the mappings coincide. Thus these ciphers are no more secure than F. The block cipher sharing scheme is no more secure than any one of the block ciphers corresponding to $F_{\{s'_j \,:\, P_j \in B\}, P_i}$ as the unauthorised set B can use an attack on $F_{\{s'_j \,:\, P_j \in B\}, P_i}$ to attack the block cipher sharing scheme since the participants in B may ignore the (additional) information they may have from knowledge of partial computations. Ideally, the block cipher sharing scheme should be no weaker than any of these block ciphers which in turn should be no weaker than F. This latter condition could be achieved if the ciphers $F_{\{s'_j \,:\, P_j \in B\}, P_i}$ were equivalent to (copies of) F.

In the next few sections we look at some block cipher sharing schemes and consider whether and in what sense they have the property that the block cipher sharing scheme is as secure as the block cipher itself.

4 Block cipher sharing based on cascades

In [7] a technique for sharing block ciphers was proposed that is based on the composition of block ciphers using the technique of cascading. We will revisit this scheme by generalising the results in [7] and suggesting a method for making it more efficient. Note that throughout this (and the next) section we will consider only cascading the same block cipher. See Section 6.3 for a discussion of the implications of composing different block ciphers in a block cipher sharing scheme.

4.1 Block cipher sharing using sequence sharing

In [7] a technique known as *sequence sharing* was introduced and used for block cipher sharing. If $E : K \times M \to C$ is a block cipher with $|M| = |C|$ so that each mapping E_k is a permutation then we may identify the elements of C with those of M via a one-to-one correspondence and define the h-fold composition (or *cascade*) of E as the block cipher $E^h : K^h \times M \to C$ given by $E^h(k_1, \ldots, k_h, m) = E(k_h, E(k_{h-1}, \ldots, E(k_1, m)\ldots))$. It is such h-fold compositions of ciphers that are shared using sequence sharing.

Let Γ be an access structure defined on the recipient set $\mathcal{P} = \{P_1, \ldots, P_n\}$. Informally, a *sequence sharing scheme* for Γ is a method of sharing a sequence of secret keys $\mathbf{k} = (k_1, \ldots, k_{m(\Gamma)})$ (where $m(\Gamma) \geq n$) by issuing each receiver in $\mathcal{P}$ with a subsequence of components of $\mathbf{k}$ in such a way that if and only if a subset of receivers $A \in \Gamma$ does A collectively hold all component keys and hence know the whole of $\mathbf{k}$.

If we have a sequence sharing scheme for Γ then it is easy to construct a block cipher sharing with respect to an $m(\Gamma)$-fold composition of a cipher and Γ. For shared decryption, the sender is issued with $\mathbf{k}$ and each recipient receives their subsequence of components of $\mathbf{k}$ under the sequence sharing scheme. The sender encrypts the message m by the $m(\Gamma)$-fold composition of the block cipher, using each component of $\mathbf{k}$ in turn. Hence the ciphertext is

$$c = E_{\mathbf{k}}^{m(\Gamma)}(m) = E_{k_{m(\Gamma)}}(E_{k_{m(\Gamma)-1}}(\cdots E_{k_1}(m)\cdots)). \tag{1}$$

The encrypted message is then passed around the recipients belonging to an authorised subset A, who sequentially strip off layers of encryption using component keys that are known to them. Thus shared decryption is performed in $r = m(\Gamma)$ steps and, for $j = 1, \ldots, r$, the partial computation $E_{k_j}^{-1}(b)$, where b is the output of the previous round, is a decryption by some participant $P_i \in A$ who holds key k_j in their share s_i. The definition of a sequence sharing scheme ensures that only an authorised set of recipients holds all the keys $k_1, \ldots, k_r$ that enable them to extract m from $E_{\mathbf{k}}^{m(\Gamma)}(m)$ in this manner. We refer to this general technique as *cascade block cipher sharing*.

Shared encryption works in a very similar way. The main advantage of cascading over the simple solution to shared encryption described in Section 2.2 is that there is no message expansion when cascading, with only one ciphertext needing to be transmitted to the receiver.

The authors of [7] observed that it is important to minimise $m(\Gamma)$ in order to maximise the efficiency of (1) and to reduce the storage of key material for each recipient. In [7] a lower bound on $m(\Gamma)$ is proven and it is demonstrated that this lower bound is optimal for three special classes of access structure.

4.2 Cumulative maps

We now show that the concept of sequence sharing has already been investigated under a different name, and that existing results both generalise and unify the various results proven in [7].

Cumulative maps were first defined in [27] but the idea behind them was first exhibited in [20]. Let Γ be an access structure on $\mathcal{P}$. We define a *cumulative map* (f, S) for Γ to be a finite set S and a mapping $f\colon \mathcal{P} \to 2^S$ from the set of participants to the set of all subsets of S such that for $A \subseteq \mathcal{P}$, we have $\bigcup_{P \in A} f(P) = S$ if and only if $A \in \Gamma$.

In [27] the following result is shown:

Result 4.1 *Let (f, S) be a cumulative map for Γ then $|S| \geq |\Gamma^+|$, where Γ^+ is the set of all maximal unauthorised sets with respect to Γ.*

A cumulative map (f, S) for Γ with $|S| = |\Gamma^+|$ is referred to in [21] as *minimal.* A minimal cumulative map for any Γ can be constructed as follows [20]. Let $S = \{U_1, U_2, \ldots, U_s\}$ be the sets of Γ^+. For each $P \in \mathcal{P}$, let $f(P) = \{U_i \mid P \notin U_i, 1 \leq i \leq s\}$. It is further proven in [21] for any Γ that the above minimal cumulative map is effectively unique, and that any (not minimal) cumulative map for Γ "contains" the minimal cumulative map in a natural sense.

Example 4.1 Let Γ be an access structure defined on the set $\mathcal{P} = \{a, b, c, d\}$ and let $\Gamma^- = \{\{a, b, c\}, \{b, d\}, \{c, d\}\}$ be the collection of the minimal authorised subsets in Γ. The minimal cumulative map (f, S) for Γ is defined as follows: $S = \{U_1, U_2, U_3, U_4\}$, where $U_1 = \{b, c\}$, $U_2 = \{a, c\}$, $U_3 = \{a, d\}$ and $U_4 = \{a, b\}$; $f(a) = \{U_1\}$, $f(b) = \{U_2, U_3\}$, $f(c) = \{U_3, U_4\}$ and $f(d) = \{U_1, U_2, U_3\}$.

We now observe that a sequence sharing scheme for Γ is nothing more than a cumulative map for Γ where there exists a publicly known ordering of the elements in the set S. Thus a cumulative map can be used in exactly the same way as in Section 4.1 for block cipher sharing. We thus have the following immediate consequence:

Theorem 4.1 *In any sequence sharing scheme for Γ the number of components $m(\Gamma)$ is such that $m(\Gamma) \geq |\Gamma^+|$. Further, there exists an optimal sequence sharing scheme for Γ with $m(\Gamma) = |\Gamma^+|$.*

Proof The bound follows immediately from Result 4.1. The optimal sequence sharing scheme is established by using the minimal cumulative map for Γ. □

We note that the bound in Theorem 4.1 generalises several bounds in [7]. Further, the three optimal constructions exhibited in [7] are all special cases of applying the minimal cumulative map to the respective access structures.

4.3 Cascade block cipher sharing using generalised cumulative maps

The technique described in Section 4.2 can be implemented optimally, with respect to the total number of component keys and number of components held by each receiver, if a minimal cumulative map is used. However, for many access structures the minimal cumulative map still involves a very large number of components. For example, in the case of a (t, n)-threshold access structure, the total number of component keys is $\binom{n}{t-1}$ and each receiver would need to hold $\binom{n-1}{t-1}$ component keys. We now propose a refinement of this technique that allows these values to be considerably reduced, at the expense of some expansion of the encrypted message.

Let Γ be an access structure on $\mathcal{P}$. We define a *generalised cumulative map* $(f_1, \dots, f_\ell, S_1, \dots, S_\ell)$ for Γ to be a collection of finite sets $S_1, \dots, S_\ell$, where each S_i is associated with a mapping $f_i\colon \mathcal{P} \to 2^{S_i}$, such that for $A \subseteq \mathcal{P}$ we have

$$\bigcup_{P \in A} f_i(P) = S_i \text{ for some } i \ (1 \leq i \leq \ell), \text{ if and only if } A \in \Gamma.$$

The application of generalised cumulative map to sharing block cipher is straightforward. Essentially, implementing a generalised cumulative map involves implementing the cumulative map solution independently ℓ times.

The cost of doing this for shared decryption is that the communication cost is increased ℓ times. There will be a piece of ciphertext corresponding to each set S_i, $i = 1, \dots, l$. However, we note that for shared encryption this is not so, since only the communication that relates to the set S_i that can be accumulated by the sending set is actually sent.

This observation also relates to the security of the shared block cipher. In the case of shared decryption, l encryptions of the same message using the same product cipher with independent keys are transmitted. However, as before, in the case of a cascade cipher E^n, this is as secure as transmitting l encryptions of the same message using the block cipher E with independent keys.

The gain is that it may be possible to (significantly) reduce both the numbers of keys for each user and the total number of keys, as well as some of the computational cost. For $\ell = 1$, a generalised cumulative map is a cumulative map as defined in Section 4.2. The question is, if $\ell \geq 2$, can the bound of Theorem 4.1 be improved?

We now show that for the case where Γ is a threshold access structure the answer is in the affirmative. Let $X = \{1, \dots, n\}$, $Y = \{1, \dots t\}$ and $\mathcal{H}$ be a set of functions from X to Y. We say that $\mathcal{H}$ is a *minimal perfect hash family* if for any subset $A \subseteq X$ with $|A| = t$ there exists an element $h \in \mathcal{H}$ such that h restricted to A is one-to-one. When $|\mathcal{H}| = \ell$ we refer to the triple $(X, Y, \mathcal{H})$ as a $PHF(\ell; n, t, t)$. Perfect hash families are interesting combinatorial objects that have found numerious applications to cryptography (see, for example, [25, 28, 29, 30]).

Given a $PHF(\ell; n, t, t)$, $(X, Y, \mathcal{H})$, we can construct a generalised cumulative array for the (t, n)-threhold structure on $\mathcal{P} = \{P_1, \dots, P_n\}$ in the following way. Let $\mathcal{H} = \{h_1, \dots, h_\ell\}$. For each h_i we associate a t-set $A_i = \{a_1^{(i)}, \dots, a_t^{(i)}\}$ and define a function f_i from $\mathcal{P}$ to A_i such that $f_i(P_j) = a_{h_i(j)}^{(i)}$. For any t subset of $\mathcal{P}$, there exists a j such that h_i restricted to it is one-to-one. However, less than t participants in $\mathcal{P}$ together are associated with at most $t-1$ elements in A_i for all $1 \leq i \leq \ell$. (Note that each participant will get only one element from each A_i). Thus $(f_1, \dots, f_\ell, A_1, \dots, A_\ell)$ is a generalised cumulative array for the (t, n)-threshold structure defined on $\mathcal{P}$.

If $\ell = 1$ then we know that $\sum_{i=1}^{\ell} |A_i| = |A_1| \geq \binom{n}{t-1}$. However, from [25] we have the following result:

Result 4.2 *For any integer ℓ, n, t, there exists a $PHF(\ell; n, t, t)$, provided $\ell \geq \lceil te^t \log n \rceil$.*

Thus, from Result 4.2, when t is small (compared to n) and fixed the value of $\sum_{i=1}^{\ell} |A_i|$ can be reduced from $\binom{n}{t-1}$ to $O(\log n)$.

Many questions arise from this construction. From a theoretical viewpoint, given ℓ, what is the lower bound on $\sum_{i=1}^{\ell} |A_i|$? Do better constructions for generalised cumulative maps for threshold access structures exist? We note that the construction of generalised cumulative maps based on PHFs is quite restrictive in the sense that each participant has only one element from A_i, but in the definition of generalised cumulative map, each participant is allowed to have multiple elements from A_i. How do we go about constructing generalised cumulative maps for general access structures? From a more practical viewpoint it is worth observing that the balance between the cases $\ell = 1$ and $\ell > 1$ suggests that there are some trade-offs between communication cost and secret key storage. How can these trade-offs be quantified and assessed?

4.4 Security of cascade block cipher sharing

When the keys of a key sequence are independent, the block cipher E^h is a cascade cipher as discussed by Maurer and Massey [24]. They argue that an attack on a cascade cipher to retrieve the message from the ciphertext can be turned into an attack on the first cipher in the cascade by embedding the first cipher in a cascade whose second and subsequent component keys are known. That is they show that the cascade cipher E^h is as secure as the cipher E. Now, E may be similarly embedded in a block cipher sharing scheme for E^h. Thus the cascade block cipher sharing of E^h when the first component key is unknown is as secure as E. Thus cascade block cipher sharing of E^h is as secure as E against an unauthorised set of participants who do not have the first component key. We state this as a theorem.

Theorem 4.2 *Let E be a block cipher. Then the security of the cascade block cipher sharing of E^h against unauthorised sets of participants who do not have the first component key is at least that of the block cipher E.*

Note that each partial computation f_j of the cascade block cipher sharing of E^h is an application of the block cipher E and each block cipher $E^h_{\{s'_j \,:\, P_j \in B\}, P_i}$ is a cascade block cipher in which some of the component keys are known. The inputs to these ciphers are the results of transformations, depending on the known component keys, applied to the message of the shared cascade cipher E^h. Thus the probability distribution on the inputs need not be the same as the probability distribution on the message space and the Maurer and Massey result does not apply. However, under the assumption that E is secure for all probability distributions on the message space (or, at least those that might possibly arise by transformations of the given distribution), the above argument again shows that cascade block cipher sharing of E^h is as secure as the block cipher E also when the first component key (and, possibly, subsequent but not all component keys) is known. We now have the following theorem.

Theorem 4.3 *Let E be a block cipher and suppose that the security of E is independent of the probability distribution on the message space. Then the security of the cascade block cipher sharing of E^h is at least that of the block cipher E.*

When the sharing of block cipher E^h provides public access to the result of the first application of the cipher E (under key k_1) an attack on E provides an attack on the shared block cipher E^h. So the shared block cipher E^h is no more secure than E. In this case the security of the sharing of block cipher E^h is equivalent to that of E since, given that a subset of participants have used their keys to produce the result of the first application of the cipher E from the ciphertext, the only subsets containing these participants that are unauthorised are those that do not know k_1.

Brickell *et al* [7] consider the security of their sequence sharing technique. They consider a random cipher E, so that the keys of a key sequence pick one permutation from a family of permutations chosen independently and with uniform probability from the set of all permutations of the message space. Each component key picks a permutation from the same family. Thus in a technical sense the keys of the key sequence are not independent as they relate to the same family of permutations and Maurer and Massey's result does not apply. Thus, in Maurer and Massey's terminology, Brickell *et al* analyse a product cipher rather than a cascade cipher. If the keys of the key sequence determined independent random ciphers (*ie* independent families of permutations are chosen for each round) then Maurer and Massey's result would apply.

Brickell *et al*'s security model gives the adversary access to two oracles. One provides the encryption of a chosen message or the decryption of a chosen ciphertext for a choice of permutation from the family and the other either encrypts a chosen message or decrypts a chosen ciphertext of the shared product block cipher E^h under the unknown key sequence. Thus this adversary does not have access to the intermediate, partial computations that the participants of an authorised subset calculate, but is provided with information about images of permutations of a family that contains those applied in these partial computations.

Brickell et al [7] bound the probability with which an adversary can predict a message-ciphertext pair of the shared product cipher E^h using a bound of Aiello *et al* [1] for the probability with which the adversary can distinguish between E^h (with some component keys known) and the random cipher E to show that the sequence sharing of the random cipher is essentially as secure as the random cipher.

Thus Brickell *et al* extend the result of Maurer and Massey to one special case where the keys of the key sequence are not independent. In view of Maurer and Massey's comments on the difference between product ciphers and cascade ciphers it seems unlikely that Brickell *et al*'s result can be extended to the case where E is a given arbitrary block cipher and the keys of the sequence are not independent.

We note that in sequence sharing for shared encryption only the first participant inputs the message to the first cipher in the product. The other participants must trust that the output that is sent to them does indeed correspond to the corrrect input. Similarly, in sequence sharing for shared decryption only the last participant's decryption outputs the message and the other participants must trust that the correct output is shared with them. Of course, if there is more than one participant holding the first (or last, or indeed any) key of the sequence then some check may be made that these participants are behaving correctly.

5 Block cipher sharing using XOR

In this section we look at an alternative composition construction for block cipher sharing. We use the same cumulative map approach to distribute component keys amongst an access structure of receivers as used in Section 4. However, rather than cascading, we use a different composition technique based on XOR.

5.1 XOR block cipher sharing

Maurer and Massey [24] have observed that a cascade cipher for which the component ciphers commute is as secure as any one (and therefore the most secure) of the components. In particular a cascade of stream ciphers is as secure as the most secure component stream cipher. This observation provides the motivation for a natural alternative to the cascade composition of ciphers considered in the previous section that uses a composition based on XOR. For this reason we refer to it as *XOR block cipher sharing.*

We assume that a cumulative map for access structure Γ has been used to distribute component keys of $\mathbf{k} = (k_1, \ldots, k_{m(\Gamma)})$ to the group of senders. Let $E : M \times K \to C$ be a block cipher. The shared encryption of message m under key sequence $\mathbf{k}$ is given by

$$E_{\mathbf{k}}^{\oplus m(\Gamma)}(m) = (m \oplus E_{k_1}^{-1}(r) \oplus E_{k_2}^{-1}(r) \oplus \cdots \oplus E_{k_{m(\Gamma)}}^{-1}(r), \;\; r),$$

where r is a random nonce or counter.

If we define a block cipher $E^{\oplus} : (M \times C) \times K \to M \times C$ by $E_k^{\oplus}(m, r) = (m \oplus E_k^{-1}(r), r)$. The block cipher we share is the $m(\Gamma)$-fold composition of $E^{\oplus}$ under the restriction that the component r input to $(E^{\oplus})^{m(\Gamma)}$ is a random nonce or counter.

The following properties of this construction are worth noting:

1. Since r is transmitted in the clear, this construction has the significant advantage over cascading that the order in which the component keys are used does not matter. Indeed the mappings $E_{k_1}^{\oplus}, E_{k_2}^{\oplus}$ commute for all keys $k_1, k_2 \in K$. This means that the encryptions to calculate the values $E_{k_i}(r)$ can, for example, be computed in parallel. For shared encryption this is an even more significant advantage since in this case the values $E_{k_i}(r)$ can even be precomputed before the message is known if there is some agreement on the generation of the nonce r.
2. The only cost of XOR block cipher sharing is an effective "doubling" of the size of the message resulting from the need to send the nonce r. Note that r must indeed be a "nonce", since re-use of r allows anyone with knowledge of a previous message/ciphertext pair to calculate $E_{k_1}(r) \oplus E_{k_2}(r) \oplus \cdots \oplus E_{k_{m(\Gamma)}}(r)$ and so determine the relation between a new message/ciphertext pair
3. For shared encryption, the ability to precompute the components and the reduced message expansion of XOR block cipher sharing are both advantages over the simple solution described in Section 2.2.
4. The generalised cumulative map techniques described in Section 4.3 can also be applied to XOR block cipher sharing. These reduce the number of component keys held by the distributed entities, whilst increasing the size of the transmitted message.

5.2 Security of XOR block cipher sharing

Since the cipher $E^{\oplus m(\Gamma)}$ is the h-fold composition of commuting ciphers the result of Maurer and Massey [24] shows that when the component keys are independent, $E^{\oplus m(\Gamma)}$ is as secure as $E^{\oplus}$. Moreover we do not have to make any assumption about the security of E with respect to the probability distribution on M nor about which component keys are unkown to an adversary. Of course, the cipher $E^{\oplus}$ sends the random nonce or counter r, in clear. However, the other part, the message m, is as secure as a decryption $E_k^{-1}(r)$ of r. Thus the sharing of block cipher $E^{\oplus m(\Gamma)}$ is as secure as the block cipher E.

The algorithm that performs the shared encryption of $E^{\oplus m(\Gamma)}$ given in the previous section is as follows. The first partial computation is $f_1(k_1, m) = (m \oplus E_{k_1}^{-1}(r), r)$ where r is a nonce. Subsequently, for $j = 2, \ldots, m(\Gamma)$, $f_j(k_j, b_0, \ldots, b_{j-1}) = (b'_{j-1} \oplus E_{k_j}^{-1}(r), r)$ where $b_{j-1} = (b'_{j-1}, r)$. Decryption is the same process (with c replacing m). Thus the sharing of $E^{\oplus m(\Gamma)}$ in which the intermediate results of partial computations are public provides an attacker with pairs $E_{k_j}^{-1}(r), r$ for keys $k_j \neq k_1$ but only the value $m \oplus E_{k_1}^{-1}(r)$. Hence, information is obtained about m if and only if information is obtained about $E_{k_1}^{-1}(r)$. Thus, in this case the security of the sharing of $E^{\oplus m(\Gamma)}$ is equivalent to that of the block cipher E.

6 Block cipher sharing based on plaintext sharing

In this section we consider a third technique for sharing a block cipher, based on directly sharing the message. As noted in Section 2.3, this technique requires the additional assumption of the existence of a local dealer. This dealer might be one of the distributed entities themselves and is trusted to either fairly generate and distribute shares of the message (for shared encryption) or to pool the message shares to reconstruct the message (for shared decryption). See Section 2.3 for comments on this requirement. However we note here that if a linear secret sharing scheme is used then the dealer does not need to know the message. Each participant receives a share of the ‘zero’of the linear scheme from the dealer and and adds it to the message to determine their share of the message. In this scenario each participant has responsibility for the inputting of the message m to their encryption so that no single participant need be trusted with the message as is the case for block cipher sharing using sequence sharing.

We present *plaintext sharing* as a legitimate technique in its own right, particularly for shared decryption. We also demonstrate how plaintext sharing can be combined with other techniques to create schemes with other interesting properties, including the ability to use different block ciphers within the shared encryption/decryption process.

6.1 Basic plaintext sharing

Basic plaintext sharing is very simple. Once again we describe the technique from the perspective of shared decryption for a block cipher E. We assume that each of the receivers $P_i, i = 1, \ldots, n$, shares a secret key k_i with the sender, and that there is a receiving access structure Γ defined on the receivers. To send a ciphertext, the sender first generates shares $m_1, \ldots, m_n$ of message m according to Γ and then encrypts each m_i using key k_i. The sender then sends the component ciphertext $E_{k_i}(m_i)$ to P_i. Any set of receivers in Γ can now decrypt their own ciphertext component and then pool their shares (via the local dealer) to recover the message.

The advantage of plaintext sharing is that if the block cipher provides sufficient computational security then the unconditional security of the plaintext sharing can mean that no information is obtained about the message m by a computationally bounded attacker, despite his obtaining (limited) partial information about the shares m_i. Nevertheless if $D \subseteq M \times S_1 \times \ldots, \times S_n$ is a secret sharing scheme for Γ on $\mathcal{P}$ and $\mathbf{d} = (m, m_1, \ldots, m_n) \in D$ then an attacker who determines information about $m_1, \ldots, m_n$ from $E_{k_1}(m_1), \ldots, E_{k_n}(m_n)$ determines information about $\mathbf{d}$ and hence possibly (but not necessarily) about m. However the information an attacker may obtain about m is no more than n times the information he obtains about the share m_i for which he obtains the maximum amount of information.

We remark further that all techniques for extending and strengthening basic secret sharing can be applied (for example using verifiable secret sharing, such as that proposed in [16], to introduce robustness).

Compared to the composition techniques however, drawbacks of basic plaintext sharing are the need for a local dealer and the message expansion resulting from having to send n component ciphertexts. However, note that for shared encryption this can be reduced to t component ciphertexts, where t is the size of a minimal set in the sending access structure (this is identical to the message expansion of the simple solution for shared encryption described in Section 2.2). Note that the message expansion in plaintext sharing will be larger than just comparing the number of component ciphertexts in the case that the access structure is not ideal.

6.2 Efficient plaintext sharing

We now show that it is possible to reduce the amount of message expansion involved in shared decryption. In doing so, we are also able to reduce the amount of component key material that the sender needs to hold, which consists of n component keys in basic plaintext sharing. These gains are made as part of a trade-off, the cost of which is an increase in the amount of component key material held by each receiver. Using the following technique we reduce the number of component keys from n to v whenever there is an appropriate access structure on v elements for sharing the message m. There is also a corresponding reduction to the message expansion.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of participants and let Γ be an access structure defined on $\mathcal{P}$. In basic plaintext sharing we shared a message m according to Γ. However, now we consider sharing m according to an access structure $\mathcal{F}$ defined on a set $X = \{x_1, \ldots, x_v\}$ of size $|X| = v < n$. We say that the pair of access structures Γ and $\mathcal{F}$ are plaintext sharing compatible if there is a subset $I \subseteq \mathcal{P} \times X$ such that, for all $A \subseteq \mathcal{P}$,

$$\{x_j \mid (P_i, x_j) \in I \text{ for some } P_i \in A\} \in \mathcal{F} \iff A \in \Gamma.$$

Given a pair of compatible access structures Γ and $\mathcal{F}$, we implement shared encryption by plaintext sharing as follows. The sender generates v keys $\{k_1, \ldots, k_v\}$ and distributes the keys $\{k_j \mid (P_i, x_j) \in I\}$ to participant $P_i, i = 1, \ldots, n$. To send message m the sender generates shares $m_1, \ldots, m_v$ of m according to $\mathcal{F}$ and transmits $E_{k_1}(m_1), \ldots, E_{k_v}(m_v)$ using the block cipher E. A subset A of participants can decrypt sufficient ciphertexts, using the keys they hold, to recover m if and only if $\{x_j \mid (P_i, x_j) \in I$ for some $P_i \in A\} \in \mathcal{F}$, that is $A \in \Gamma$.

It is easily seen that basic plaintext sharing is the case $X = \mathcal{P}, \mathcal{F} = \Gamma$ and $(P, x) \in I$ if and only if $x = P$. Also, if $f : \mathcal{P} \to 2^X$ is a cumulative map for the access structure Γ then $\mathcal{F} = \{X\}$ is a compatible access structure with corresponding set $I = \{(P, x) \in \mathcal{P} \times X \mid x \in f(P)\}$ since a subset $A \subseteq \mathcal{P}$ holds all the keys if and only if $\cup_{P \in A} f(P) = X$.

As for basic plaintext sharing, the computational security of the block cipher together with the unconditional security of the sharing of the message can mean that no information about the message m is obtained by an attacker of limited resources.

An interesting special case arises when Γ is a threshold access structure. We show Γ may be plaintext sharing compatible with another threshold access structure. We need the following concept defined in [14]. A set system $(X, \mathcal{B})$ with $X = \{x_1, \ldots, x_v\}$ and $\mathcal{B} = \{B_i \mid B_i \subseteq X, i = 1, \ldots, n\}$ is called a (v, n, t)-*strong cover-free family* (or v, n, t)-SCFF) if for any $\Delta, \Lambda \subseteq \{1, \ldots, n\}$ with $|\Delta| = t$ and $|\Lambda| = t - 1$,

$$|\cup_{i \in \Delta} B_i| > |\cup_{j \in \Lambda} B_j| \, .$$

We now show how to construct a block cipher sharing scheme using a SCFF. Let $(X, \mathcal{B})$ be a (v, n, t)-SCFF. We construct a (t, n)-shared decryption scheme as follows:

- The sender chooses a v-set of keys $K = \{k_1, \ldots, k_v\}$ and gives receiver P_i the subset $A_i = \{k_j \mid \text{for all } j \text{ with } x_j \in B_i\}$. Then $(K, \{A_1, \ldots, A_n\})$ is a (v, n, t)-SCFF.
- Let
$$u = \min_{\Delta \subseteq \{1, \ldots, n\}, |\Delta| = t} |\cup_{j \in \Delta} A_j| \, .$$
For a message m, the sender breaks m into v shares $m_1, \ldots, m_v$, using a (u, v)-threshold scheme.
- The sender encrypts the v shares using each of the v component keys from K and sends $E_{k_1}(m_1), \ldots, E_{k_v}(m_v)$ to the receivers.
- Any t receivers, pooling their keys, can decrypt at least u components $E_{k_i}(m_i)$ and so can recover the message m.

It is fairly straightforward to verify that the above construction indeed results in a (t, n)-shared decryption scheme.

The efficiency of the (t, n)-shared decryption scheme is completely determined by the parameters of the (v, n, t)-SCFF. Note that in this scheme the numbers of component keys for the sender and receiver P_i are v and $|B_i|$ respectively, and the message expansion is given by v. To reduce this message expansion compared to basic plaintext sharing we need to use a (v, n, t)-SCFF in which, for given t and n, the value v is as small as possible. Several constructions for such SCFFs are given in [14]. In particular it was shown that for any fixed t there exists an infinite class of (v, n, t)-SCFFs in which $v = O(\log n)$.

6.3 A combined scheme for sharing block ciphers

We have seen that we were limited in how we could apply the result of Maurer and Massey [24] to cascade block cipher sharing. We now show that this drawback may be compensated for by combining cascade block cipher sharing with plaintext sharing. To be able to use the result of [24] as in Section 4.4, and thus claim that the scheme is as secure as the first component block cipher, we need to force each receiver to become involved in the decryption of the first component. We can achieve this by combining the cascade technique with plaintext sharing, as follows.

Let E be a secure block cipher. Let Γ be an access structure and put $h = m(\Gamma)$. Suppose that the h-fold composition E^h of block cipher E is to be shared according to access structure Γ. Suppose that $\mathbf{k_1} = (k_1, \ldots, k_h)$ is a corresponding sequence of component keys. For $i = 2, \ldots, h$ we put $\mathbf{k_i} = (k_i, \ldots, k_h, k_1, \ldots, k_{i-1})$. We use the (h, h)-threshold scheme to share a message m as shares $m_1, \ldots, m_h$ and encrypt m as $E^h_{\mathbf{k_1}}(m_1), \ldots, E^h_{\mathbf{k_h}}(m_h)$. We call this *combined plaintext-cascade block cipher sharing.*

We observe that each key component is used as the first key in the cascade encryption of one message share. Thus, since an unauthorised subset does not have at least one of the key components, at least one of the message shares is secure by theorem 4.2. Since the message is shared according to a (h, h)-threshold scheme the message itself is also secure.

We note that this method can be extended to a composition $E_h \circ E_{h-1} \circ \ldots \circ E_1(\mathbf{k_1}, m) = E_h(k_h, E_{h-1}(k_{h-1}, \ldots, E_1(k_1, m)$ of different block ciphers (this is a natural extension of the basic problem discussed in Section 4). When using the cascade technique of Section 4 we cannot now rely alone on the result of [24] for the overall security of the block cipher sharing scheme since not all ciphers applied in the cascade need provide the same level of security. However, using the combined plaintext-cascade block cipher sharing technique and encrypting the message share m_i using the composition of $E_1, \ldots, E_h$ in cyclic order beginning with E_i, each component cipher is used as the first cipher applied to one message share. It follows that the security provided to the message m is that of the strongest of the ciphers $E_1, \ldots, E_h$ (even if their relative strengths are unknown).

The cost of the guarantee that the block cipher sharing scheme is as secure as the strongest component block cipher is a h-fold message expansion. We note that this can be reduced by using generalised cumulative maps in a similar way to the application in Section 4.3.

7 Conclusions and extensions

In this paper we have continued the investigation started in [7] concerning the use of symmetric cryptographic primitives in threshold cryptgraphy. We have looked at a number of techniques for sharing the encryption or decryption of a block cipher, and commented on their relative values. We believe that this topic merits further examination, particularly with respect to increasing the robustness and trying to reduce the message expansion involved in some of current proposals.

We conclude by briefly outlining two natural extensions of the concept of block cipher sharing considered thus far.

7.1 Multiple senders and multiple receivers

We have only described our block cipher sharing schemes in the context of either a single sender communicating with a distributed receiver, or a distributed sender communicating with a single receiver. It is possible to distribute both ends of the communication. To do this we associate one access structure Γ_S with the sending end, and another access structure Γ_R with the receiving end. The goal of the block cipher sharing scheme is now to send an encrypted message so that:

1. The block cipher sharing scheme must be at least as secure as using a single application of the block cipher to encrypt the message.

2. After attempting to decrypt the ciphertext the receiver is able to determine whether the ciphertext was encrypted by a group of senders in Γ_S or not.

3. A group of receivers can decrypt the ciphertext if and only if they belong to Γ_R.

Firstly we consider the two composition constructions, based on cascades and XOR. These two constructions use the idea of a cumulative map. Consider the simplest case, where the schemes are based on adoption of the minimal cumulative map. Suppose that the minimal cumulative maps associated with Γ_S and Γ_R are (f_S, T_S) and (f_R, T_R) respectively. If $|T_S| = |T_R|$ then we can apply the schemes as already described, where each cumulative map is defined on the same set of component keys. The senders use (f_S, T_S) to jointly encrypt the message, and the receivers use (f_R, T_R) to jointly decrypt the ciphertext.

Otherwise, without loss of generality suppose that $|T_S| < |T_R|$. All that we need to do to extend the basic schemes as already described is to extend the minimal cumulative map (f_S, T_S) so that it is defined on T_R. This can trivially be done by defining a new mapping f'_S on the set of senders, where for each sender P, $f'_S(P) = f(P) \cup T$, where $|T| = |T_R \setminus T_S|$. We can now implement either of the composition constructions as already described, using the (not minimal) cumulative map (f'_S, T_R) at the sending end, and (f_R, T_R) at the receiving end. Note that this particular extension of the minimal cumulative map (f_S, T_S) might not be the most efficient one as it increases the number of component keys that each sender must hold. More work is required to identify more efficient extensions for use in this type of example. A similar extension applies if generalised cumulative maps are used for either of the composition constructions.

Plaintext sharing can also be extended to a completely distributed environment. Again suppose that the associated access structures are Γ_S and Γ_R. The senders initially receive shares of the message based on access structure Γ_S. The techniques of [13, 22] can then be used to restribute these shares to the access structure Γ_R. By following one of these redistribution protocols, the senders can send information to the receivers that allows only receivers in Γ_R to recover the message. As well as recovering the message, these receivers will also be able to determine that a set of senders in Γ_S were involved in the encryption, since the receivers will recover all the subshares used in the redistribution (see [13, 22] for details).

7.2 Sharing other symmetric primitives

Although we have concentrated on sharing block ciphers in this paper, many of these techniques could be used to share the computation of other symmetric cryptographic primitives.

For example, all of the techniques that we describe can be used directly to share the computation of a stream cipher. In this case we simply replace the block cipher encryption (decryption) with stream cipher encryption (decryption) and proceed in an identical manner.

A more interesting case is that of sharing a message authentication code (or MAC). In the analogue of shared decryption, a single sender wants to send a message to a group of receivers so that a group of receivers can only determine the data origin and integrity of the message if the group belong to the access structure associated with the receiving end of the communication. The direct analogue of plaintext sharing would certainly work in this case. (The sender breaks up the message into shares, computes a different MAC on each share, and sends these MACs off to the receivers associated with each key. The integrity of the shares can only be verified if the MACs are checked on a collection of shares that determine the whole message - in other words a set of shares that belong to the receiving access structure.)

Since a MAC is normally a compression function, there is no direct analogue of the cascade construction for MACs. However, a simplified version of XOR block cipher sharing would seem to work for sharing a MAC. It suffices that a cumulative map (f, T) is used to distribute component keys of the (receiving) access structure Γ_R and that the sender computes:

$$\mathrm{MAC}_{\mathbf{k}}(m) = \mathrm{MAC}_{k_1}(m) \oplus \mathrm{MAC}_{k_2}(m) \oplus \cdots \oplus \mathrm{MAC}_{k_{|T|}}(m).$$

Only a group of receivers belonging to Γ_R can compute all $|T|$ of the "component MACs" and hence verify the MAC that was sent by the sender. Initial work on sharing MACs using this type of construction has recently been investigated in [23].

References

[1] B. Aiello, M. Bellare, G. Di Crescenzo and R. Venkatesan. Security Amplification by Composition: the case of Doubly-Iterated, Ideal Ciphers, *Advances in Cryptology: Crypto '98, Lecture Notes in Computer Science*, 1462 (1998) 390–407.

[2] S.R. Blackburn. Combinatorics and Threshold Cryptography. In *Combinatorial Designs and their Applications, Chapman and Hall/CRC Research Notes in Mathematics*, 403, F.C. Holroyd, K.A.S. Quinn, C. Rowley and B.S. Web (Eds.), CRC Press, London (1999) 49–70.

[3] S.R. Blackburn and P.R. Wild. Optimal linear perfect hash families. *J. Comb.Theory - Series A*, 83 (1998) 233–250.

[4] D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys, *Advances in Cryptology: Crypto '97, Lecture Notes in Computer Science*, 1233 (1997) 425–439.

[5] C. Boyd. Digital Multisignatures. In *Cryptography and Coding*, Oxford University Press (1989) 241–246.

[6] E.F. Brickell and D.M. Davenport. On the classification of ideal secret sharing schemes. *J. of Cryptology*, 4 (1991)123–134.

[7] E. Brickell, G. Di Crescenzo and Y. Frankel. Sharing Block Ciphers, *Information Security and Privacy, Lecture Notes in Computer Science*, 1841 (2000) 457–470.

[8] R. Canetti and S. Goldwasser. An Efficient Threshold Public-Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack, *Advances in Cryptology: Eurocrypt '99, Lecture Notes in Computer Science*, 1592 (1999) 90–106.

[9] D. Chaum and E. van Heyst. Group Signatures, *Advances in Cryptology: Eurocrypt '91, Lecture Notes in Computer Science*, 547 (1991) 257–265.

[10] C. Cocks. Split Knowledge Generation of RSA Parameters. *Cryptography and Coding, Lecture Notes in Computer Science*, 1355 (1997) 89–95.

[11] A. De Santis, Y. Desmedt, Y. Frankel and M. Yung. How to Share a Function Securely, *Proceedings of ACM Symp. Theory of Computing (STOC) '94* (1994) 522–533.

[12] Y. Desmedt and Y. Frankel. Threshold Cryptosystems, *Advances in Cryptology: Crypto '89, Lecture Notes in Computer Science*, 435 (1990) 307–315.

[13] Y. Desmedt and S. Jajodia. Redistributing Secret Shares to New Access Structures and its Applications, *Technical Report ISSE TR-97-01*, George Mason University, July 1997.

[14] Y. Desmedt, R. Safavi-Naini and H. Wang. Redistribution of Mechanical Secret Shares, *Financial Cryptography'02*, to appear.

[15] S. Even and O. Goldreich. On the Power of Cascade Ciphers, *ACM Transactions on Computer Systems,* 3 (1985) 108-116.

[16] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, IEEE Press (1987) 427–437.

[17] P.-A. Fouque, G. Poupard and J. Stern. Sharing Decryption in the Context of Voting or Lotteries, *Financial Cryptography 2000, Lecture Notes in Computer Science*, 1962 (2001) 90–104.

[18] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust Threshold DSS Signatures, *Advances in Cryptology: Eurocrypt '96, Lecture Notes in Computer Science*, 1070 (1996) 354–371.

[19] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust and efficient sharing of RSA functions, J. of Cryptology, 13(2) (2000) 273–300.

[20] M. Ito, A. Saito and T. Nishizeki. Secret Sharing Scheme Realizing General Access Structure. *J. Cryptology*, 6 (1993) 15–20.

[21] W.-A. Jackson and K.M. Martin. Cumulative Arrays and Geometric Secret Sharing Schemes, *Advances in Cryptology: Auscrypt '92, Lecture Notes in Computer Science*, 718 (1993) 48–55.

[22] K.M. Martin, R. Safavi-Naini and H. Wang. Bounds and Constructions for the Redistribution of Secret Shares to New Access Structures, *The Computer Journal*, 42(8) (1999) 638–649.

[23] K.M. Martin, J. Pieprzyk, R. Safavi-Naini, H. Wang and P. Wild. Threshold MACs, *5th International Conference on Information Security and Cryptology: ICISC'02.* Novermber 2002, Seoul, Korea, LNCS 2587.

[24] U. Maurer and J. Massey. Cascade Ciphers: The Importance of Being First, *Journal of Cryptology*, Vol. 6, No.1 (1993) 55–61.

[25] K. Mehlhorn. **Data Structures and Algorithms**, Volume 1, Springer-Verlag, 1984.

[26] A. Shamir. How to Share a Secret, *Communications of the ACM*, 22 (1976), 612–613.

[27] G.J. Simmons, W.-A. Jackson and K. Martin. The Geometry of Shared Secret Schemes, *Bulletin of the ICA*, 1 (1991), 71–88.

[28] D. R. Stinson, T. van Trung and R. Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures, *J. Statist. Plan. Infer.,* **86**(2000), 595-617.

[29] D. S. Stinson, R. Wei and L. Zhu. New constructions for perfect hash families and related structures using combinatorial designs and codes, *J. Combin. Designs*, **8**(2000), 189-200.

[30] H. Wang and C. Xing. Explicit Constructions of perfect hash families from algebraic curves over finite fields. Journal of Combinatorial Theory - Series A, **93**(2001), 112-124.