Sheaf Tools for Encryption

Elena Şendroiu
Université Paris 7 - Denis Diderot

Abstract

This paper proposes the use of sheaf theory in security encryption in parallel and distributed computing systems. So, we exploit the architecture of distributed computing systems. In particular, we are using a general notion of sheaf as a functor on a category with a Grothendieck topology, i.e. a site. Sheaf properties give data reconstitution. This can be used in distributed encryption and authentication. In particular, our sheaf definitions provide computation tools to accomplish these goals, hence we have made optimal procedures for construction and validation of sheaves. Thereafter, we describe an encryption algorithm and an authentication method by sheaves. Moreover, we have introduced a new notion in sheaf theory on a site, family for matching, in order to simplify the sheaf definition more. This has given us the possibility to construct an optimal encryption algorithm by sheaves. In addition, the generated matching family, from a family for matching, is used to verify data integrity.

1 Introduction

First note that, before the doctoral studies the author worked in computer security research (anti-virus research, software protections, cryptography and authentication).

This paper proposes the use of sheaf theory in security encryption in parallel and distributed computing systems. In particular, we are using a general notion of sheaf as a functor on a category $\mathscr{C}$ equipped with a Grothendieck topology J, i.e. a site $(\mathscr{C}, J)$. These notions are explained in appendix A.

It results from sheaf properties the following relation:

a value is in correspondence with a set of values distributed on certain points.

This is the relation that we propose to be used in distributed encryption. It follows that we can exploit the architecture of distributed computing systems. Moreover, a complex architecture gives a richness of bijections. Thereafter, we are beginning to connect parallel and distributed computing systems with sheaf theory. First of all, the base category $\mathscr{C}$, used in our modelling, is generated by a network. The objects are network nodes and the arrows are connections between nodes. More clearly, this is the associated category to the network's graph. We will see, in this paper, that all computation tools are building from the base category that is provided by the architecture of distributed computing systems. So, the following section titled *Framework* presents how we can compute sheaf tools for accomplishing our goals.

2 Framework

As necessary background, appendix A presents a short synthesis of sheaf theory on a site. More detailed explanations are in [2, 4]. Here, we give some ideas about this. For example, we say only that a Grothendieck topology (def. A.5) is a collection of sieves that satisfy some axioms.

2.1 Sieves

Definition 2.1 *Given a category $\mathscr{C}$. Given an object C of $\mathscr{C}$, a **sieve** on C is a family S of arrows with codomain C such that: if $f \in S$ and the composition $f \circ h$ is defined*

then $f \circ h \in S$*, i.e.* sieve rule. A sieve on a point (node) C is thus a family of arrows with the same codomain C and is closed under left composition. For example the sieve presented in figure 1, is generated by the arrows $f \circ p$ and h. By the sieve rule, this sieve contains also the arrows $f \circ p \circ q$ and $h \circ r$.

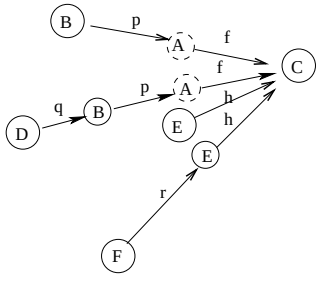

Figure 1:

So, a sieve is a modelling of any data-processing connection. Since it may have many arrows between two nodes and back arrows then a sieve has a richer and more interesting structure than a tree. More precisely, a sieve is a kind of extension, because the old structure is preserved. Moreover, it is up graded by the sieve rule which provides a new structure.

Thereafter we can determine the collection of sieves on any node from the architecture of systems.

2.2 Generating Procedure of Sites

Definition 2.2 *A* ***site*** $(\mathscr{C}, J)$ *is a pair consisting of a small category* $\mathscr{C}$ *and a Grothendieck topology* J *on* $\mathscr{C}$. A Grothendieck topology J (def. A.5, app. A) is a collection of sieves that satisfy some axioms. In particular, a sieve of a Grothendieck topology is called covering sieve. The use of sites is necessary for defining the sheaves on a site. Since, the verification of the axioms of a Grothendieck topology is algorithmisable, in [4, 6] we have described an algorithm for generating Grothendieck topologies from the base category. In particular, in our modellings, the base category corresponds to the network's graph. More exactly, from the architecture of distributed computing systems we can generate all possible Grothendieck topologies on this as a base category. In other words, sieves, and in particular, sites are given from the architecture of systems.

Then we can suppose a Grothendieck topology, generated by the algorithm described in [4, 6], on the base category.

2.3 Validation and Construction of Sheaves

A presheaf P on $\mathscr{C}$ is a functor $P : \mathscr{C}^{op} \longrightarrow$ *Sets*. The collection of presheaves on $\mathscr{C}$ gives the category of the presheaves $Sets.^{\mathscr{C}^{op}}$ Any presheaf is a colimit of a diagram of representable functors [2].

Definition 2.3 *We call representable functor on point* C *the functor* $R = y(C) = Hom_{\mathscr{C}}(-, C) : \mathscr{C}^{op} \longrightarrow$ *Sets, where* $y : \mathscr{C} \longrightarrow Sets^{\mathscr{C}^{op}}$ *is the Yoneda embedding [2].* Hence $R(D)$ is the set of all arrows from node D to node C. For any $f : E \longrightarrow D$, $R(f) : R(D) \longrightarrow R(E)$ is defined by $R(f)(g) = f \circ g$. This results also from the architecture of systems. Moreover, any diagram of representable functors corresponds to a diagram of the base category (lemma A.4, app. A).

Consequently a presheaf is provided clearly starting from the architecture of systems.

A sheaf F for a Grothendieck topology J is a presheaf which satisfies the sheaf condition on the site $(\mathscr{C}, J)$ (def. A.8, app. A). The category of the sheaves $Sh(\mathscr{C}, J)$ on a site $(\mathscr{C}, J)$ is a subcategory of the category of the presheaves $Sets.^{\mathscr{C}^{op}}$ We know that a presheaf can be generated from the base category. Then, by using the associated sheaf functor [2, 4] we can associate a sheaf to any presheaf P for a certain Grothendieck topology J. Thereafter, we have given a construction algorithm of sheaves in [4, 6]. The validation process supposes the verification of the sheaf condition (def. A.8.c). To performing this, we have used first our sheaf definition provided by prop. A.10. Moreover, in order to writing an optimal validation algorithm of sheaves, we have introduced the concept of family for matching (def. A.13, app. A) that allows to reformulate the sheaf definition A.8. In addition, by using the notion of families for matching we have defined in [4] another associated sheaf functor (th. A.24). This allows us to write an algorithm that generates efficiently sheaves.

Consequently a sheaf is also provided starting from diagrams of the base category that is determined by the architecture of systems.

2.4 Reconstitution and Data Integrity

By definition A.8.a, a matching family of a sheaf $\mathcal{O}$ on a site $(\mathscr{C}, J)$, for a covering sieve S of $C \in \mathscr{C}$, is denoted by $\{x_f \in \mathcal{O}(dom(f))|\ x_f \cdot g = x_{fg},\ \ \forall_{g \in f^*(S)}\}_{f \in S}$, where $x_f \cdot g = \mathcal{O}(g)(x_f)$. The collection $\bigcup\{x_f / x_f \in \mathcal{O}(dom(f))\}_{f \in S}$ is as a set of possible results, these results are partial or final for each execution generated by the arrows of the cover S. If this results set satisfies the local aspect, i.e. all x_{fg} are given locally by x_f by applying it the procedure $\mathcal{O}(g)$, then there is a unique list of initial attributes (locally input), called amalgamation (def. A.8.b), which determines all the executions. Thereafter, if there are consistent partial informations then we can reconstitute all the process.

Consequently, it follows that this can be applied to verifying data integrity in parallel and distributed computing systems.

3 Security of Systems by Sheaves

In this section, we exploit the sheaf condition on a site. So, sheaf properties give data reconstitution. This can be used in distributed encryption and authentication. The base category, used in our modelling, is generated by a network. The objects are network nodes and the arrows are connections between nodes. More clearly, this is the associated category to the network's graph. Then we can suppose a Grothendieck topology, generated by the algorithm described in [4, 6], on the base category. In addition, our sheaf definitions provide computation tools to accomplish this goal. In fact, section 2.3 presents how can compute sheaf tools. So, we have seen that we may generate sheaves for this process, in accordance with [4, 6], starting from diagrams of the base category. Note that, from a large network one can generate a big sheaf i.e. one can make choices from several diagrams of the base category.

3.1 Encryption by Sheaves

We know that a matching family of a sheaf $\mathcal{O}$ on a site $(\mathscr{C}, J)$, for a covering sieve S of $C \in \mathscr{C}$, is denoted by $\{x_f \in \mathcal{O}(dom(f))|\ x_f \cdot g = x_{fg},\ \ \forall_{g \in f^*(S)}\}_{f \in S}$ (def. A.8, app. A). Section 2.4 interprets it as a set of partial or final possible results. These results are produced from executions generated by arrows of S. By the sheaf condition (def. A.8 app. A), every matching family is generated by a unique amalgamation $x \in \mathcal{O}(C)$. Moreover, there is a bijection between them, expressed by

$$Nat(S, \mathcal{O}) \cong Nat(y(C), \mathcal{O}) \cong \mathcal{O}(C), \qquad (1)$$

where the sieve S is regarded as a subfunctor of $y(C)$ (prop. A.9, app. A). Consequently, sheaf properties give data reconstitution useful in computation.

More precisely, a sheaf has the property that there exists a unique amalgamation (an element x in a set) for any matching family of any cover of any object in the category. The element (amalgamation) x can be thought as a seed which gives rise to a unique set of elements obtained by means of a sieve and distributed in the network nodes (the matching family corresponds in fact to this set).

Thereafter, we propose the use of sheaf theory in distributed encryption and authentication. By definition, encryption is the translation of data into a secret code (plaintext $\mapsto$ ciphertext). Encryption is the most effective way to achieve data security. The reverse of encryption is called decryption (ciphertext $\mapsto$ plaintext).

So, the encryption process, by a sheaf $\mathcal{O}$, is done by steps in conformity with the procedures $\mathcal{O}(f)$ for all f of a cover S. Note that we must choose some cover S such that $\mathcal{O}(f)$ **has not a reasonable complexity** for any $f \in S$. This encryption is adapted to distributed systems. We can also use disjoint subsheaves (prop. A.21) to encrypt differently. Since a distributed encryption by a sheaf corresponds to a unique input by the sheaf universal property, we can find a decryption of it. More detailed explanations of this idea are given in sections 3.2, 3.3 and 3.5.

By using the concept of family for matching (def. A.13, app. A), which we have introduced, our lemma A.18 simplifies the sheaf condition that is expressed by

$$\mathcal{O}(C) \cong \{< x_i > \in \prod_{i \in I} \mathcal{O}(dom(f_i))|\ Comp(x_i, f_i)\}, \quad (2)$$

$$Comp(x_i, f_i) = \forall_{i,j \in I}(f_i \circ g = f_j \circ h) \Rightarrow (x_i \cdot g = x_j \cdot h)$$

and $\{f_i : D_i \longrightarrow C|\ i \in I\}$ is a minimal covering family, such as any arrow does not factorize itself by another, which generates a covering sieve S. This is a bijection between amalgamations and families for matching. In addition, any family for matching generates a

matching family of the same way that a minimal covering family generates a covering sieve and by using matching family condition. If there is no restriction provided by the compatibility condition (2) then there are only free sites hence the sheaf condition is a simpler bijection $\mathcal{O}(C) \cong \prod_{i \in I} \mathcal{O}(dom(f_i))$.

For example the sieve presented in figure 1, sec. 2.1, is generated by the arrows $f \circ p$ and h. By the sieve rule, this sieve contains also the arrows $f \circ p \circ q$ and $h \circ r$. So, the corresponding minimal cover family of the sieve S is the arrows $f \circ p$ and h. Given a family for matching $\{x_{fp}, x_h\}$ distributed in the corresponding points B, E. Then the matching family, generated from this family for matching, is $\{x_{fp}, x_{fpq}, x_h, x_{hr}\}$ distributed in the corresponding points B, D, E, F, where $x_{fp} \cdot q = x_{fpq}$ and $x_h \cdot r = x_{hr}$. In [4], we have proof that any amalgamation of such a family for matching is also an amalgamation of the generated matching family and reciprocally. It is for all these reasons that we must promote the use of families for matching.

The public key can be the network's graph (or even arrows of the covering sieves). Since a family for matching allows less computations, the secret key can be a minimal covering family (or even the used Grothendieck topology). In addition, the generated matching family (from a family for matching) is used to verify data integrity.

Moreover, the matching families are generated to divert intruders into traps. So, our ciphertext is given by families for matching as certain parts of encrypted data. This manner of selecting certain parts of encrypted data by secret key in the decryption process provides difficult obstacles to intruders.

3.2 Encryption Algorithm Description

Now, we present an encryption algorithm by sheaves. So, our proposal is to assign the elements of $\mathcal{O}(C)$, for a given node C, to the characters (i.e. secret codes). If the set $\mathcal{O}(C)$ is too large we can thus assign many codes to characters. It is recommended that at least the more frequent characters have several codes. For this assignation, we can use a classical algorithm with secret key for the following operation: *List of characters* $\mapsto \mathcal{O}(C)$.

In addition, we can share the subsheaves on groups of characters, in particular a subsheaf contains the more frequent characters. Then these codes will be encrypted by the sheaf $\mathcal{O}$ or certain subsheaves of $\mathcal{O}$ as sets of another codes distributed on the network.

For example the character "A" is first coded by $x_a \in \mathcal{O}(C)$, hence, by using a minimal covering family $\{f_i | \, i \in I\}$, its encryption is $\{x_a \cdot f_i | \; i \in I\}$, where $x_a \cdot f_i = \mathcal{O}(f_i)(x_a)$. Since the collection $\{x_a \cdot f_i | \; i \in I\}$ has the amalgamation x_a, from lemma A.16, app. A, it is a family for matching. This is distributed on the network i.e. each $x_a \cdot f_i$ is put in the corresponding node $\mathcal{O}(dom(f_i))$. In particular the character "A" is a frequent character hence it has, in addition, another codings.

An interesting idea is to assign the elements of $\mathcal{O}(C)$ to some strings of characters or even words of the plaintext. For example: $and \mapsto \alpha_1$, $now \mapsto \alpha_2$, ...

Generally, a set of the form $\{x_j \in \mathcal{O}(C) | \; j \in J\}$ is encrypted in the following set of sets $\{\{x_j \cdot f_i \in \mathcal{O}(dom(f_i)) | \; j \in J\} | \; i \in I\}$ that distributed on the network i.e. any node $\mathcal{O}(dom(f_i))$, where $i \in I$, contains a set of the form $\{x_j \cdot f_i | \; j \in J\}$.

By sheaf condition, any set, that satisfies the compatibility condition (2), of the form $\{x_{f_i} | \; i \in I\} \in \prod_{i \in I} \mathcal{O}(dom(f_i))$, where $\{f_i : D_i \longrightarrow C | \; i \in I\}$ is a minimal covering family, corresponds to a unique code $x \in \mathcal{O}(C)$. This allows the decryption of sets as $\{x_{f_i} | \; i \in I\}$ distributed on the nodes $\{dom(f_i) | \; i \in I\}$. For example, in figure 2, the plaintext $abcde$ is encrypted in two distributed ciphertexts $a'b'c'd'e'$ and $a''b''c''d''e''$. By sheaf condition, we will obtain the value a from the pair (a', a''), b from (b', b''), etc.

Moreover, we put random data or false messages on transparent nodes of covering sieves (i.e. some arrows of their minimal covering families can factorize in these nodes by arrows which are not in corresponding covering sieves) to produce false tracks for adversaries. We can also put random data on other covers of network, other than secret covers, in order to grow uncertainty. In addition, this random data is an encryption of other random data. These encryptions are produced especially by other sheaves or subsheaves of the used sheaf. Consequently, in this case adversaries are in a growing uncertainty.

Thereafter, the secret key can be: the codes of characters (or the secret key of the assign algorithm), a minimal

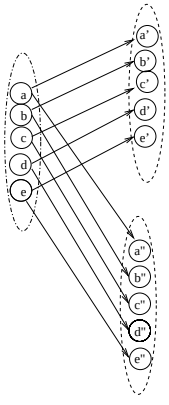

Figure 2: *Distributed encryption*

covering family (or even the used Grothendieck topology), the sheaf (or the diagram of the base category which generates this sheaf). The public key can be the network's graph (or even arrows of the covering sieves).

The clear meanings of the secret keys in the protocols are the nodes which contain the real ciphertext. By using a public key which is some arrows of a covering sieve we can determine the secret nodes from the used Grothendieck topology that is a secret key.

In addition, the encrypted data recovery can be done by using the authentication method by sheaves described in section 3.7.

3.3 Decryption

The decryption of encrypted data (by our method) is done in the following way:

1. If the minimal covering family, that gives the nodes which contain the our ciphertext, is not fixed (known as secret key) then we determine it by using the public key which is some arrows of the corresponding covering sieve;

2. Next we can select parts of the encrypted data to obtain our ciphertext;

3. We can now check the data integrity by using the matching family condition since encrypted data are matching families generated from our ciphertext;

4. We obtain the plaintext by considering that there is a unique amalgamation and performing the reverse of assign process i.e. $\mathcal{O}(C) \mapsto$ *List of characters*.

There are two strategies that achieve goal 4: (I) computing codes for a long ciphertext or (II) searching amalgamations for a short ciphertext. We compute as in sheaf encryption process, by using the secret key of assign algorithm, the sets of encrypted codes that correspond to the *List of characters*. Then we can decrypt our ciphertext distributed in the secret nodes. Note that we can determine only the codes of certain characters (eg, the more frequent) and in the case when these are not sufficient to decrypt the ciphertext then we seek another codes.

3.4 Intrusion Detection by Sheaves

Note that, by sheaf universal properties we can detect if the ciphertext is modified since we cannot obtain a unique amalgamation. Thereafter, we cannot perform the decryption process if the ciphertext is modified by an attacker or falsified. This is in addition an intrusion detection system by sheaves.

3.5 An example

For example we perform the plaintext

I am here.

First we assign the elements of $\mathcal{O}(C)$ to the characters of this plaintext, i.e. $I \mapsto x_I$, " " $\mapsto x_{bl_1}$, $a \mapsto x_a$, $m \mapsto x_m$, " " $\mapsto x_{bl_2}$, $h \mapsto x_h$, $e \mapsto x_{e_1}$, $r \mapsto x_r$, $e \mapsto x_{e_2}$, where x_I, x_{bl_1}, x_a, x_m, x_{bl_2}, x_h, x_{e_1}, x_r, x_{e_2} are in $\mathcal{O}(C)$. Note that, since are more frequent the characters " " (i.e. space) and e have two codes.

For encryption, we use the corresponding minimal cover family of the sieve S, figure 1, that is the arrows $f \circ p$ and h. So, the set

$$\{x_I,\ x_{bl_1},\ x_a,\ x_m,\ x_{bl_2},\ x_h,\ x_{e_1},\ x_r,\ x_{e_2}\}$$

is encrypted in the following set of sets (i.e. a set of families for matching)

$\mathcal{C} = \{\{x_I \cdot (f \circ p),\ x_I \cdot h\},\ \{x_{bl_1} \cdot (f \circ p),\ x_{bl_1} \cdot h\},\ \{x_a \cdot (f \circ p),\ x_a \cdot h\},\ \{x_m \cdot (f \circ p),\ x_m \cdot h\},\ \{x_{bl_2} \cdot (f \circ p),\ x_{bl_2} \cdot h\},\ \{x_h \cdot (f \circ p),\ x_h \cdot h\},\ \{x_{e_1} \cdot (f \circ p),\ x_{e_1} \cdot$

$h\}$, $\{x_r \cdot (f \circ p),\ x_r \cdot h\}$, $\{x_{e_2} \cdot (f \circ p),\ x_{e_2} \cdot h\}\}$ that is our ciphertext and is distributed on the secret nodes B and E.

So, the node B contains the set $\{x_I \cdot (f \circ p),\ x_{bl_1} \cdot (f \circ p),\ x_a \cdot (f \circ p),\ x_m \cdot (f \circ p),\ x_{bl_2} \cdot (f \circ p),\ x_h \cdot (f \circ p),\ x_{e_1} \cdot (f \circ p),\ x_r \cdot (f \circ p),\ x_{e_2} \cdot (f \circ p)\}$ and the node E contains the set $\{x_I \cdot h,\ x_{bl_1} \cdot h,\ x_a \cdot h,\ x_m \cdot h,\ x_{bl_2} \cdot h,\ x_h \cdot h,\ x_{e_1} \cdot h,\ x_r \cdot h,\ x_{e_2} \cdot h\}$.

Note that, we use here only families for matching. In addition, the generated matching family (from a family for matching) is used to verify data integrity. So, the set $\{x_I,\ x_{bl_1},\ x_a,\ x_m,\ x_{bl_2},\ x_h,\ x_{e_1},\ x_r,\ x_{e_2}\}$ is encrypted in the following set of sets (i.e. a set of matching families)

$\mathcal{M} = \{\{x_I \cdot (f \circ p),\ \{x_I \cdot (f \circ p \circ q),\ x_I \cdot h,\ x_I \cdot (h \circ r)\},\ \{x_{bl_1} \cdot (f \circ p),\ \{x_{bl_1} \cdot (f \circ p \circ q),\ x_{bl_1} \cdot h,\ x_{bl_1} \cdot (h \circ r)\},\ \{x_a \cdot (f \circ p),\ x_a \cdot (f \circ p \circ q),\ x_a \cdot h,\ x_a \cdot (h \circ r)\},\ \{x_m \cdot (f \circ p),\ x_m \cdot (f \circ p \circ q),\ x_m \cdot h,\ x_m \cdot (h \circ r)\},\ \{x_{bl_2} \cdot (f \circ p),\ \{x_{bl_2} \cdot (f \circ p \circ q),\ x_{bl_2} \cdot h,\ x_{bl_2} \cdot (h \circ r)\},\ \{x_h \cdot (f \circ p),\ x_h \cdot (f \circ p \circ q),\ x_h \cdot h,\ x_h \cdot (h \circ r)\},\ \{x_{e_1} \cdot (f \circ p),\ x_{e_1} \cdot (f \circ p \circ q),\ x_{e_1} \cdot h,\ x_{e_1} \cdot (h \circ r)\},\ \{x_r \cdot (f \circ p),\ x_r \cdot (f \circ p \circ q),\ x_r \cdot h,\ x_r \cdot (h \circ r)\},\ \{x_{e_2} \cdot (f \circ p),\ x_{e_2} \cdot (f \circ p \circ q),\ x_{e_2} \cdot h,\ x_{e_2} \cdot (h \circ r)\}\}$ distributed in the corresponding points B, D, E and F.

In the decryption operation, we must extract $\mathcal{C}$ from $\mathcal{M}$, more precisely from the secret nodes B and E, in order to verify first data integrity.

Since an amalgamation of a family for matching is also an amalgamation of the generated matching family and reciprocally, we are using the set $\mathcal{C}$ in the decryption process. In this case, we have a short ciphertext. So we will search amalgamations, i.e. we will use strategy II. In decryption the ciphertext $\mathcal{C}$ is seen as $\{\{x_{I,fp},\ x_{I,h}\},\ \{x_{bl_1,fp},\ x_{bl_1,h}\},\ \{x_{a,fp},\ x_{a,h}\},\ \{x_{m,fp},\ x_{m,h}\},\ \{x_{bl_2,fp},\ x_{bl_2,h}\},\ \{x_{h,fp},\ x_{h,h}\},\ \{x_{e_1,fp},\ x_{e_1,h}\},\ \{x_{r,fp},\ x_{r,h}\},\ \{x_{e_2,fp},\ x_{e_2,h}\}\}$.

By the sheaf condition there is a unique amalgamation x_k such that $x_k \cdot (f \circ p) = x_{k,fp}$ and $x_k \cdot h = x_{k,h}$ for any k in $\{I,\ bl_1,\ a,\ m,\ bl_2,\ h,\ e_1,\ r,\ e_2\}$. For finding the amalgamations

$$\{x_I,\ x_{bl_1},\ x_a,\ x_m,\ x_{bl_2},\ x_h,\ x_{e_1},\ x_r,\ x_{e_2}\}$$

we search some x in $\mathcal{O}(C)$ until all these amalgamations are founded.

In this case, the node A is transparent. In other words, the arrow f is not in the sieve S, but S contains the arrows fp and fpq that are factorized by f. Hence fp is in the minimal covering family of S. So, we can put in this node random data or false messages to divert intruders into traps.

3.6 Discussion

By our concept of family for matching, the sheaf properties are concretized in an explosion of bijective mappings between a set and a cartesian product of several sets. This richness of bijections generated by sheaf theory from a network architecture can be thus used in encryption. Fortunately, we ourself have discovered the notion of family for matching. If not, i.e. if this algorithm handles only matching families then has a weakness. It follows that the expert adversaries could broke it if they will discover this notion of family for matching and the secret keys.

It is recommended that the texts of big size should be shared in many parts in order to encrypt on different covers on the network. In this case, we may choice some covers where the minimal covering family contains only an arrow and thus the ciphertext will have then the same size as the plaintext.

In addition, a software built in accordance with our ideas can provide a real example. So, it remains for us to try this out.

3.7 Authentication by Sheaves

The system safety must ensure the data integrity and confidentiality. The access control accomplishes them by using an authentication system.

In particular, the richness of bijections generated by sheaf theory from a network architecture can be used in authentication. So, the correspondence (1) between matching families and amalgamations in sheaf theory can be used in password authentication [4, 5, 7]. Thereafter, like in the preceding section, the base category is generated by a network, i.e. the associated category to the network's graph. In addition, we may also generate covering sieves by an algorithm described in [4, 6] and sheaves in accordance with [4] for this process.

Now, we present our method of authentication by sheaves. So, a password is an amalgamation, hence a matching family for a covering sieve corresponds to some

encryptions of the password on distributed network nodes. In other words, a password correspond to a data set distributed on network. For example, given a sieve S (fig. 1, sec. 2.1) which contains the following arrows $f \circ p$, $f \circ p \circ q$, h, $h \circ r$. Encryptions of a password x are $x_{fp}, x_{fpq}, x_h, x_{hr}$ distributed in the corresponding points B, D, E, F, where $x_{fp} \cdot q = x_{fpq}$ and $x_h \cdot r = x_{hr}$. Note that, the computations x_{hr} and x_{fpq} are parallel.

The bijection (1) between matching families and amalgamations allows to realize the authentication process. Therefore, if x is a password of a user and the matching family is kept by the authentication system, then the authentication process just checks whether or not x gives rise to the same family of matching. For this example, there is a unique password x that corresponds to the set $\{x_{fp}, x_{fpq}, x_h, x_{hr}\}$, distributed in the corresponding points B, D, E, F, such that $x \cdot k = x_k$, for all $k \in \{fp, fpq, h, hr\}$. More precisely, this is a kind of re-authentication.

Note that, our sheaf definition (prop. A.10, app. A) provides us matching families for this process and expresses the unique amalgamation by $M(1)$, where M is a matching functor.

The role of the sieves is to indicate the access rights. Moreover, the use of a covering sieve led to determine an access cover in a work environment. From the compatibility property of a matching family, i.e. local aspect (all x_{fg} are given locally by x_f by applying it the procedure $\mathcal{O}(g)$ as in section 2.4), two equivalent paths allow the same access.

Consequently, the authentication process must be on certain covering sieves in which the authentication method **has not a reasonable complexity**.

4 Conclusions and Future Work

Sheaf properties give data reconstitution. This can be used in distributed encryption and authentication. Thereafter, we have proposed an encryption algorithm by sheaves. So, we have began to connect parallel and distributed computing systems with sheaf theory. It follows that we can exploit the architecture of distributed computing systems. In addition, we have seen that all computation tools are building from the base category that is provided by the architecture of distributed computing systems. Our approach is based on the fact that a sieve is a modelling of any data-processing connection. Also, we can determine the collection of sieves on any node from the architecture of systems, hence we can generate Grothendieck topologies (sites).

Consequently, the richness of bijections generated by sheaf theory from a network architecture can be used in encryption and authentication. In addition, our sheaf definitions provide computation tools to accomplish these goals. Since this method is based on a high mathematical framework, well adapted to distributed systems, it could be solved better than it is done by other formal frameworks.

Note that we can do trap activities. So, ciphertext is given by families for matching and the matching families are generated to divert intruders into traps. This manner of selecting certain parts of encrypted data by secret key in the decryption process provides difficult obstacles to intruders. Moreover, we can put in certain nodes random data to divert also intruders into traps.

It remains for us to try these out. Only, a software built in accordance with our ideas can provide real examples. We plan to implement this in the future.

References

[1] C. Kaufman, R. Perlman and M. Spencer: Network Security, Private Communication in Public World, Pretince Hall, (1995)

[2] Saunders Mac Lane and Ieke Moerdijk: Sheaves in Geometry and Logic, A First Introduction to Topos Theory. Springer Verlag (1991)

[3] W. Stallings: Cryptography and Network Security: Principles and Practice, Second Edition, Pretince Hall, (1999)

[4] Elena Şendroiu: Topos, un modèle pour l'informatique. PhD thesis. Université Paris 7, (2004)

[5] Elena Şendroiu: From anti-virus to sheaf tools for security. In U.E. Gattiker (Ed), EICAR 2004 Conference CD-rom: Other Contributions (ISBN: 67-987271-6-8) 18 pages. Copenhagen: EICAR e.V.

[6] Elena Şendroiu: Sheaf Algorithms, SYNASC 2004 Conference, (Ed. Mirton, ISBN: 973-661-441-7) Timişoara

[7] Elena Şendroiu: Sheaf Tools in Network Security, IASTED-PDCS 2004 Conference: Cambrige, MA, USA

A Topology of Grothendieck and Sheaves

An open U of a topological space X is identified with the unique monomorphism $U \longrightarrow X$. In a more general category, we replace monomorphisms by more general arrows $C \longrightarrow D$.

Definition A.1 *The functor $Q : \mathscr{C}^{op} \longrightarrow Sets$ is a subfunctor of the functor $P : \mathscr{C}^{op} \longrightarrow Sets$ if $QC \subseteq PC$ for all C and each $Q(f) : QC \longrightarrow QD$ is a restriction of $P(f)$ for all arrows $f : D \longrightarrow C$ of $\mathscr{C}$. Thus Q is a subobject of P.*

Remark A.2 *If $Q \subset Hom_{\mathscr{C}}(-, C)$ is a subfunctor then the set $S = \{f |\ f \in Q(dom(f))\}$ is clearly a sieve on C. Hence, we can consider a sieve on C as a subfunctor of $Hom_{\mathscr{C}}(-, C)$.*

Definition A.3 *If S is a sieve on C and $h : D \longrightarrow C$ is any arrow with codomain C then $h^*(S) = \{g | cod(g) = D$ and $hg \in S\}$ is a sieve on D.* This is a change operation of the sieve base.

Lemma A.4 *Any diagram of representable functors corresponds to a diagram of the base category.*

Demonstration. Given a small category $\mathscr{C}$ and C, D in $\mathscr{C}$. By Yoneda lemma it holds that

$Hom_{\widehat{\mathscr{C}}}(y(C), y(D)) \cong y(D)(C) \cong Hom_{\mathscr{C}}(C, D)$.

Definition A.5 *Given a category $\mathscr{C}$. A **topology** (of Grothendieck) on the category $\mathscr{C}$ is a function J which assigns to each object C of $\mathscr{C}$ a collection $J(C)$ of sieves on C such as*

i) the maximal sieve $t_C = \{f | cod(f) = C\}$ is in $J(C)$;

ii) stability axiom. *if $S \in J(C)$ then $h^*(S) \in J(D)$ for any arrow $h : D \longrightarrow C$;*

iii) transitivity axiom. *if $S \in J(C)$ and R is any sieve on C such that $h^*(R) \in J(D)$ for all arrows $h : D \longrightarrow C$ of S then $R \in J(C)$.*

Definition A.6 *We say that S is a **covering sieve** (cover) of C or that S covers C if $S \in J(C)$.*

Remark A.7 *In particular, an ordinary topological space is a site. So a sieve on an open U is exactly a subset $S \subseteq \mathcal{O}(U)$, where $\mathcal{O}(U) = \{V |\ V$ is open and $V \subseteq U\}$.*

Definition A.8 *Given a site $(\mathscr{C}, J)$, a presheaf P on $\mathscr{C}$ and a covering sieve S of an object C of $\mathscr{C}$. We denote $x \cdot f = P(f)(x)$, where $x \in P(dom(f))$.*

*a) A **matching family** for S of elements of P is a function which assigns to each element $f : D \longrightarrow C$ of S an element $x_f \in P(D)$ such as $x_f \cdot g = x_{fg}\ \forall\ g : E \longrightarrow D$ in $\mathscr{C}$. We denote it by $\{x_f |\ x_f \in P(domf) \wedge x_f \cdot g = x_{fg}\}_{f \in S}$.*

*b) An **amalgamation** of such a matching family is a single element $x \in P(C)$ such as $x \cdot f = x_f$ for all $f \in S$.*

*c) A presheaf P is a **sheaf** (for topology J) if every matching family of any cover of any object of $\mathscr{C}$ has a unique amalgamation i.e.* sheaf condition.

d) Separate condition is the sheaf condition with "has a unique" replaced by "has at most one".

Since a sieve S on C is regarded also as a subfunctor of yC, a matching family $f \mapsto x_f$ for $f \in S$ is the same thing as a natural transformation $\phi : S \longrightarrow P$. From this the following proposition holds by using Yoneda lemma [2].

Proposition A.9 *P is a sheaf iff for every covering sieve S on C the inclusion $S \longrightarrow yC$ induces an isomorphism $Hom(S, P) \cong Hom(yC, P) \cong P(C)$ [2, 4].*

In addition, a sieve S on C is the same thing as a subcategory of the category slice $\mathscr{C}/C$. Then we define a matching functor $M \in Sets^{S^{op}}$ for a presheaf P by $M(f) = \{x_f\}$, where $x_f \in P(dom(f))$, and $f \circ g \longrightarrow f \mapsto \{x_f\} \longrightarrow \{x_{fg}\}$, with $x_{fg} = P(g)(x_f)$, for all $f,\ fg \in S$. This gives the following new sheaf definition.

Proposition A.10 *A presheaf P is a sheaf iff, for all covering sieves S of objects C, any matching functor $M : S^{op} \longrightarrow Sets$ has a unique extension to $\mathscr{C}/C$. In addition, $M(1_C)$ is the unique amalgamation [4].*

Remark A.11 *A matching functor provides clearly a matching family since the collection of its object values is by definition a matching family. Thus, this is a computation tool. Hence we can build a validation procedure of sheaves which verifies if any matching functor has a unique extension to $\mathscr{C}/C$.*

Next, we will introduce the concept of family for matching to simplify again the sheaf definition A.8. More precisely, this notion provides the base elements for constructing matching families by using matching family condition. So, the following lemmas will be used to writing an optimal validation algorithm of sheaves.

Definition A.12 *Given a site $(\mathscr{C}, J)$. We call minimal covering family (m.c.f.) of an object C in $\mathscr{C}$ a family of arrows $\{f_i : D_i \longrightarrow C |\ i \in I\}$, such as any arrow is not factorized by another of this family, and generates a covering sieve of C.*

Definition A.13 *Given a site $(\mathscr{C}, J)$, a presheaf P and a minimal covering family $\{f_i : D_i \longrightarrow C |\ i \in I\}$ of an object C in $\mathscr{C}$. A **family for matching** for $\{f_i |\ i \in I\}$ is a function which assigns to each arrow $f_i : D_i \longrightarrow C$ an element*

$x_{f_i} \in P(D_i)$, *such that, whenever there are the arrows* g *and* h *such that* $f_i \circ g = f_j \circ h$, *where* $i, j \in I$, *then*

$$x_{f_i} \cdot g = x_{f_j} \cdot h \qquad (2')$$

Remark A.14 *Clearly, a family for matching* $\{x_{f_i} | \ i \in I\}$ *for* $\{f_i | \ i \in I\}$ *of elements of* P *is every element of* $\{< y_i > \in \prod_{i \in I} P(dom(f_i)) | \ \forall_{i,j \in I}(f_i \circ g = f_j \circ h) \Rightarrow (y_i \cdot g = y_j \cdot h)\}$.

Remark A.15 *Any family for matching generates a matching family in the same way that a minimal covering family generates a covering sieve and by using matching family condition i.e.* $x_{f_i \circ g} = x_{f_i} \cdot g$. *Hence, any matching functor is also generated from a family for matching.*

Lemma A.16 *If an element* $\{x_{f_i} | i \in I\}$ *of* $\prod_{i \in I} P(dom(f_i))$ *has an amalgamation* x *then this element is a family for matching [4].*

Lemma A.17 *Any amalgamation of such a family for matching is also an amalgamation of the generated matching family and reciprocally [4]. A presheaf* P *is a sheaf (for topology* J*) iff any family for matching of any minimal covering family of all objects* C *has a unique amalgamation [4] .*

Lemma A.18 *A presheaf* P *is a sheaf for a covering sieve generated by a m. c. f.* $\{f_i : D_i \longrightarrow C | \ i \in I\}$ *iff for all* $x \in P(C)$ *the application* $x \mapsto \{x \cdot f_i | \ i \in I\}$ *gives the following isomorphism* $P(C) \cong \{< y_i > \in \prod_{i \in I} P(dom(f_i)) | \ \forall_{i,j \in I}(f_i \circ g = f_j \circ h) \Rightarrow (y_i \cdot g = y_j \cdot h)\}$ *[4].*

Remark A.19 *Any presheaf* P *is a sheaf for the trivial topology since, for all* C*, the minimal covering family of the maximal sieve is only* 1_C *and* $1_{P(C)}$ *is a bijection.*

A site $(\mathscr{C}, J)$ is called free if for any minimal covering family $\{f_i | \ i \in I\}$ there are no arrows g and h such that $f_i \circ g = f_j \circ h$, where $i, j \in I$. Hence, in a free site, for any m. c. f. $\{f_i | \ i \in I\}$ every element of $\prod_{i \in I} P(dom(f_i))$ is a family for matching.

Corollary A.20 *If* $(\mathscr{C}, J)$ *is a free site then a presheaf* P *is a sheaf for a covering sieve generated by a minimal covering family* $\{f_i : D_i \longrightarrow C | \ i \in I\}$ *iff for all* $x \in P(C)$ *the application* $x \mapsto \{x \cdot f_i | \ i \in I\}$ *gives the following isomorphism* $P(C) \cong \prod_{i \in I} P(dom(f_i))$.

Proposition A.21 *Let a sheaf* $E \in Sh(\mathscr{C}, J)$. *A* subsheaf A *of* E *is a subfunctor such that for all* $C \in \mathscr{C}$, $e \in E(C)$ *and for all cover* S *of* C *satisfies the* subsheaf condition *i.e. if* $e \cdot f \in A(D)$ *for all* $f : D \longrightarrow C$ *of* S *then* $e \in A(C)$ *[2].*

Given a site $(\mathscr{C}, J)$. The category of sheaves $Sh(\mathscr{C}, J)$ is a subcategory of the category of presheaves on $\mathscr{C}$. The following theorems allow us to write an algorithm to generate sheaves.

Theorem A.22 *The inclusion functor* $i : Sh(\mathscr{C}, J) \longrightarrow Sets^{\mathscr{C}^{op}}$ *is right adjoint to the associated sheaf functor* $a : Sets^{\mathscr{C}^{op}} \longrightarrow Sh(\mathscr{C}, J)$, *where* $a(P) = (P^+)^+$ *and an element of* $P^+(C)$ *is an equivalence class of matching families* $\mathsf{x} = \{x_f | \ f : D \longrightarrow C \in R\}$, $x_f \in P(D)$, *and* $x_f \cdot k = x_{fk}$, *for all* $k : E \longrightarrow D$ *where two such families* $\mathsf{x} = \{x_f | \ f \in R\}$ *and* $\mathsf{y} = \{y_g | \ g \in S\}$ *are equivalent when there is a common refinement* $T \subseteq R \cap S$ *with* $T \in J(C)$ *s. t.* $x_f = y_f \ \forall f \in T$ *[2].*

Remark A.23 *Thus, if a topology* J *is not subcanonic, i.e. representable functors are not sheaves, then we can construct the sheaf* $ay : \mathscr{C} \xrightarrow{y} \widehat{\mathscr{C}} \xrightarrow{a} Sh(\mathscr{C}, J)$. *Since any presheaf is a colimit of a diagram of representable functors [2] then* $F \cong ai(F) \cong a lim_k \ y(C_k) \cong lim_k(ay(C_k))$ *for any sheaf* F.

Consider the map $y(C) \xrightarrow{\eta} ay(C) \xrightarrow{\alpha} X$. *From the adjunction* $i \dashv a$ *and Yoneda lemma [2] it results that*

$X(C) \cong Hom(yC, iX) \cong Hom_{Sh}(ayC, X)$.

In addition, by using the new notion of families for matching (def. A.13) we have defined in [4] another associated sheaf functor a' that is also adjoint to the inclusion functor. This allows us to write an algorithm that generates efficiently sheaves.

Theorem A.24 *Another associated sheaf functor* a' *is defined by* $a'(P) = (P^*)^*$, *where an element of* $P^*(C)$ *for any* C *in* $\mathscr{C}$ *is an equivalence class of families for matching* $\mathsf{x} = \{x_{f_i} | \ i \in I\}$, *(*$\{f_i | \ i \in I\}$ *is a m.c.f.) with* $x_{f_i} \in P(dom(f_i))$, *where two such families* $\mathsf{x} = \{x_{f_i} | \ i \in I\}$ *and* $\mathsf{y} = \{y_{g_j} | \ j \in J\}$ *are equivalent when there is a m.c.f. of the form* $\{f_i \circ p_1 = g_j \circ p_2 | \ i \in I'$ *and* $j \in J'\}$, *where* $I' \subseteq I$ *and* $J' \subseteq J$, *such that* $x_{f_i} \cdot p_1 = y_{g_j} \cdot p_2$ *for all* $i \in I'$ *and* $j \in J'$ *[4].*

Lemma A.25 *Any equivalence class of matching families is generated by an equivalence class of families for matching [4].*

Moreover, there is a map of presheaves $\eta' : P \longrightarrow P^*$ defined for each $x \in P(C)$ with $\eta'_C(x) = \{x \cdot f | \ f \in m(t_C)\}$, where $m(t_C)$ contains 1_C and all another arrows of the maximal sieve on C such that are not factorized by another. For a morphism $h : C' \longrightarrow C$ in $\mathscr{C}$ the restriction map $P^*(C) \longrightarrow P^*(C')$ is given by $\{x_{f_i} | \ i \in I\} \cdot h = \{x_{hg} | \ g \in h^{\#}(\{f_i / i \in I\})\}$, where $x_{hg} = x_h \cdot g$ and $h^{\#}(\{f_i / i \in I\}) = m(t_C)$ if $\exists k \in I$ s.t. $f_k \circ p = h$ and if it does not, is $\{h_j : D_j \longrightarrow B | \ \exists i \in I \ s.t. \ hh_j = f_i\}$ [4].

Lemma A.26 *(i) A presheaf* P *is separated iff* $\eta' : P \longrightarrow P^*$ *is a monomorphism. (ii) A presheaf* P *is a sheaf iff* $\eta' : P \longrightarrow P^*$ *is an isomorphism [4].*