Secret Sharing Schemes Based on Error Correcting Codes

Jean Gasnier

24/07/2021

ABSTRACT :

This report presents an introduction to secret sharing schemes (SSSs) based on linear error correcting codes. It starts with a reminder of the basic notions of information theory and a brief presentation of error-correcting codes. Some examples are given to introduce the fundamentals of SSS, followed by formal definitions. The links between polymatroids and SSS are explained, and then the two constructions (of Shamir and Massey) of SSS based on linear codes are presented. The report ends with the presentation of applications of SSS, specifically Shamir's.

This report has been written during a 2 months internship in the ATI team of the Institut de Mathématiques de Marseille (I2M), supervised by Alexis Bonnecaze, from 05/20 to 07/23. As part of the internship, a presentation has been made at the I2M PhD students seminar (see this link).

Table des matières

Introduction

Secret sharing schemes (or SSSs) are methods for distributing a secret into several shares in such a way that only specific groups of shares can be reconstructed. They were introduced by A. Shamir [26] with a polynomial point of view, and by G. R. Blakley [2] from a linear point of view in 1979. Shamir justified the interest of the theory by giving the example of secret key storage : indeed, SSSs offer a way to reduce the risk of key loss, while reducing the risk of key disclosure after an intrusion. SSSs are still used in cloud storage services today[16]. Other applications of SSSs were then studied, in particular for secure multiparty computation [9].

From the theoretical point of view, in 1981, R.J. McEliece and D.V. Sarwate show the existence of underlying corrective codes in the Shamir SSS [21]. This link is then particularly explored in the literature. Two constructions of SSS based on linear corrector codes are studied : Shamir's construction, and Massey's construction. In 1993, J.L. Massey studies a different construction from Shamir's and introduces the notion of minimal codes, important in his construction [19].

SSS are closely related to the notion of access structure. In 1991, E.F. Brickell and D.M. Davenport made explicit the link between access structure and matroids [4]. The use of matroids is particularly useful for working on ideal SSSs and SSSs information rates, notions that are not explored here (see [30], [4], and [24] for more details).

The report is organized as follows :

In the first part of the report, we recall some results from information theory, in particular on entropy and conditional entropy. The objective of this section is to introduce some notions and results useful for the definition of SSSs in section 3, and to build the intuition around the notion of entropy.

The second part presents generalities on error-correcting codes, and introduces the Hamming and Reed-Solomon codes. This part follows the plan and essentially repeats the results of the beginning of D. Boucher's course [3]. Some parts of the course are not repeated (in particular those on cyclic and BCH codes) because the notions they introduce are not used in the rest of the report. The proofs of theorems 2.2 and 2.3 have been modified, either to detail some aspects or to highlight some facts. The remarks and the body of the section are personal work.

The third part introduces SSS with some examples (self-made unless otherwise stated). We then define the access structures, as well as the mixed adversary model [14].We give the definition of an SSS (entropy model), then we present matroids and their link with SSSs. This presentation is largely based on [24]. The proof of the theorem 3.1 has been reworked to detail some aspects, and to improve some formulations. Linear SSSs are introduced from Shamir's SSS, and the construction of Shamir's linear SSS is presented. We quote the results of [11] on the access structures of these SSSs. Theorem 3.2 is a corrected version of [11], and the proof is self-made, using a lemma mentioned in [24]. We then introduce the Massey construction from linear matroids, and we present the results of [11] on the access structures of SSSs based on the Massey construction. Theorem 3.8 is a corrected version of [11].

The final section presents important applications of Shamir's SSS and more generally of all SSSs, including secure multiparty computation and secure storage.

1 Elements of information theory

In this part, we recall some elements of information theory that we will use in part 3 of this report for secret sharing schemes. This part takes up the notions introduced by Shannon in [28] and [27], and follows the presentation made in [24].

1.1 Entropy

Let X be a random variable with values in a finite set $\mathscr{X}$, with distribution p_X. We suppose without loss of generality that for all x, $p_X(x) > 0$.

Definition 1.1. We call **entropy of** X the real

$$H(X) = \sum_x p_X(x) \log(\frac{1}{p_X(x)})$$

where *log* denotes the logarithm in base 2.

The entropy of X defined in this way allows intuitively to measure the quantity of information contained in X. We illustrate this with two examples :

Example 1.1 (Illustration)**.** Let $\mathscr{X} = North, South, East, West$. Let us suppose that a random generator draws an element of $\mathscr{X}$ at random with uniform probability. We wish to transmit the random sequence, so we need to find a way to encode the information in binary alphabet, so as to minimize the length of the data to send.

We can see quite quickly that an optimal encoding is to associate the values 00, 01, 10, 11 and the elements $North$, $South$, $East$, $West$. The average size expected to be sent to each draw is thus 2. At the same time, we calculate that the entropy of a draw is

$$sum_{x \in \mathscr{X}} \frac{1}{4} \log(4) = 4 \frac{1}{4} \log(4) = 2,$$

that is to say as much as the average size.

Example 1.2 (Non uniform case)**.** Let us now put $\mathscr{X} = \{A, B, C\}$ and suppose that the random generator draws an A or a B with probability $\frac{1}{4}$, and C with probability $1/2$. $\mathscr{X}$ has three elements so we will necessarily have some elements encoded with two bits. Moreover, if one element is encoded by 0 and another by 1, there will be ambiguity during decoding because the third element will be encoded by 1's and 0's. A solution is therefore to encode one element by 0 and the others by 10 and 11. The C element appears the most frequently so it is this one that is encoded by 0.

The average encoding size of a draw is then

$$\frac{1}{4}.2 + \frac{1}{4}.2 + \frac{1}{2} = \frac{3}{2}.$$

We check that it is still the entropy of a draw.

Remark 1.1. In the second example, the static Huffman encoding was constructed. Huffman showed that this encoding is the shortest encoding on average among the encodings where each outcome of X is associated with a static symbol, e.g. which remains the same throughout the encoding.

In these examples, the entropy gives the lower bound on the number of bits needed to represent the information. Shannon demonstrated this result in the general case. We can thus see the interest of using entropy to describe the quantity of information contained in a random variable.

Let us study some properties of entropy :

Proposition 1.1. Let X be a random variable with values in a finite set X, then :

— $0 \leqslant H(X) \leqslant \log(\mathrm{card}(\mathscr{X}))$.

— $H(X) = 0$ if and only if X is almost certainly constant.

— $H(X) = \log(\mathrm{card}(\mathscr{X}))$ if and only if X follows the uniform distribution on X.

In cryptography, entropy is often used to verify the security of a primitive. To do this, we model the secret we are studying by a random variable. It can for example be a key drawn at random according to a uniform law. The entropy then represents the information of the random variable, e.g. the information that we are missing, the uncertainty about the secret. Decreasing this entropy means decreasing the uncertainty and therefore decreasing the quality of the protocol. The previous property shows us that if the entropy falls to 0 the secret is revealed. We also understand that the uniform distribution during the random drawing of a secret is the distribution which gives the best uncertainty on the secret (only during a drawing, it is necessary to pay attention to the paradox of the birthdays, among others, in complex protocols).

1.2 Conditionnal Entropy

Let X be a random variable with values in a finite set $\mathscr{X}$, and Y a random variable with values in $\mathscr{Y}$, which follow the joint distribution p_{XY} on X. We call p_X and p_Y the marginal laws. For y, we note $p_X(\cdot|y)$ the law of X knowing $Y = y$, and we note $p_{X|Y}$ the law of X knowing Y.

We note $H(XY) = H((X,Y)) = \sum_{x\in\mathscr{X}, y\in\mathscr{Y}} p_{XY}(x,y)\log(\frac{1}{p_{XY}(x,y)})$.

Definition 1.2. Let $y \in \mathscr{Y}$. We define **the entropy of X knowing $Y = y$** as the real number

$$H(X|Y=y) = \sum_{x\in\mathscr{X}} p_X(x|y)\log(\frac{1}{p_X(x|y)}).$$

Definition 1.3. We define **the entropy of X knowing Y** as the real number

$$H(X|Y) = \sum_{y\in\mathscr{Y}} p_Y(y)H(X|Y=y).$$

The entropy of X knowing Y represents the amount of information or uncertainty remaining for X after determining Y. In particular, the difference $H(X)-H(X|Y)$ represents the amount of information that Y brings about X.

Proposition 1.2. We have the following properties :

— $0 \leqslant H(X|Y) \leqslant H(X)$.

— $H(X|Y) = 0$ if and only if for all y there exists $x \in X$ such that $p_X(x|y) = 1$.

— $H(X|Y) = H(X)$ if and only if X and Y are independent.

— $H(XY) = H(Y) + H(X|Y) = H(X) + H(Y|X)$.

In the context of cryptography, in security proofs, conditional entropy allows to quantify the information gain offered by some specific elements, for example the cipher, or a clear-known pair. In particular, X can only be recovered from Y if $H(X|Y) = 0$. However, if $H(X|Y) \neq H(X)$ then Y brings information about X (one can imagine that the distribution of X knowing Y is modified), which can have important consequences, for example reducing the computation time necessary to break the protocol.

Finally, we have one last property :

Proposition 1.3. Let Z be a random variable with values in $\mathscr{Z}$ finite. Then

$$H(X|YZ) \leqslant H(X|Y).$$

This property justifies that the knowledge of Y and Z brings more information about X than the knowledge of Y only.

2 Error Correcting Codes

Before proceeding on to secret sharing schemes, we present some elements of the theory of correcting codes. We will see in part 3 that some notions introduced in this domain are involved in the results concerning linear secret sharing schemes.

The model that we pose in the context of the study of corrective codes is the following : two entities A and B want to communicate on a channel subject to turbulence. This turbulence can create errors in the message that B receives from A. The objective of using error-correcting codes is to detect these errors and to be able to correct them under certain assumptions, limiting the addition of computation time and the congestion of the information channel. To do this, it is necessary to have encoding and decoding algorithms that are not very time consuming, and to limit the amount of supernumerary data.

More formally, the model used is *binary symmetric channel*, which means that the error probability is constant on each character sent, independently of the position. One may mention as a remark the *binary erasure channel* model, where the receiver receives either a character correctly, or an error indicator with a certain probability, again invariant. The difference is the knowledge of the position of the error.

This part is a repetition of D. Boucher's course [3]. It is essentially based on the research articles of R. W. Hamming [15] and of I. S. Reed and G. Solomon [25].

2.1 General correcting codes

We first give ourselves an alphabet $\mathscr{A}$ (e.g. a finite set) of cardinal q, and two positive non-zero integers k and n such that $k \leqslant n$.

Definition 2.1. We call $(n,k)_q$ **code** a subset C of $\mathscr{A}^n$ of cardinal q^k. We call n the **length** of the code and k its **log-cardinal**.

Remark 2.1. Any $(n,k)_q$ code C can be seen as the image of a one-to-one function ϕ of $\mathscr{A}^k$ into $\mathscr{A}^n$.

The idea of error-correcting codes is to transform a word on the alphabet $\mathscr{A}$ of length k (e.g.. an element of $\mathscr{A}^k$) into a word of length n (e.g. an element of $\mathscr{A}^n$) thanks to ϕ, then to transmit this word of length n. Some words of $\mathscr{A}^n$ are not in C, so if such a word is received, we know that there was an error during transmission.

A message of any size can be transmitted by splitting the message into pieces of size k and sending each piece individually (eventually completing the message with a null character until its length is multiple of k).

Definition 2.2. Let C be a $(n,k)_q$ code. We call the value $R = \frac{k}{n}$ the **information rate** of C.

The lower the information rate, the greater the amount of information to be sent, and the longer it takes to send the initial message. It is necessary to pay attention to the information rate when choosing a correcting code in order not to increase the sending time, the bandwidth and the transmission delay.

Definition 2.3. We call **Hamming metric** the function d of $\mathscr{A}^n \times \mathscr{A}^n \longrightarrow \mathbb{N}$ such that

$$\forall x, y \in \mathscr{A}^n, d(x,y) = \operatorname{card}\left(\{i \in [\![1,n]\!] | x_i \neq y_i\}\right).$$

Proposition 2.1. The Hamming metric is a metric.

Proof

— The symmetry immediatly follows from the definition.

— Let $x, y \in \mathscr{A}^n$. If $x \neq y$, $\{i \in [\![1, n]\!] | x_i \neq y_i\}$ is not empty, therefore $d(x, y) \neq 0$. If $x = y$, $\{i \in [\![1, n]\!] | x_i \neq y_i\}$ is empty, therefore $d(x, y) = 0$.

— Let $x, y, z \in \mathscr{A}^n$. $\{i \in [\![1, n]\!] | x_i \neq z_i\} \subset \{i \in [\![1, n]\!] | x_i \neq y_i\} \cup \{i \in [\![1, n]\!] | y_i \neq z_i\}$, so

$$\begin{aligned} d(x, z) = \text{card}(\{i \in [\![1, n]\!] | x_i \neq z_i\}) &\leqslant \text{card}(\{i \in [\![1, n]\!] | x_i \neq y_i\} \cup \{i \in [\![1, n]\!] | y_i \neq z_i\}) \\ &\leqslant \text{card}(\{i \in [\![1, n]\!] | x_i \neq y_i\}) + \text{card}(\{i \in [\![1, n]\!] | y_i \neq z_i\}) \\ &= d(x, y) + d(x, z). \end{aligned}$$

Definition 2.4. We call **minimum distance of** C the integer $d = \min_{x \neq y} d(x, y)$. From here on, we will note the code C $(n, k, d)_q$ instead of $(n, k)_q$ to take into account its minimum distance.

Proposition 2.2. Let C be a $(n, k, d)_q$ code. Let $c \in C$ and $y \in \mathscr{A}^n$, such that $d(c, y) \leqslant d - 1$. Then $y = c$ or $y \notin C$.

Proof

Suppose for the sake of argument that $y \in C$ and $y \neq c$. Then $d(c, y) < d = \min_{x \neq y} d(x, y) \leqslant d(c, y)$, which is absurd.

Proposition 2.3. Let C be a $(n, k, d)_q$ code. Let $y \in \mathscr{A}^n$, then there exist at most one $c \in C$ such that $d(c, y) \leqslant \left\lfloor \frac{d-1}{2} \right\rfloor$.

Proof

This follows from the triangle inequality.

Definition 2.5. We call **correction rate** the integer $\tau = \left\lfloor \frac{d-1}{2} \right\rfloor$.

Example 2.1 (Repetition code)**.** Let $\mathscr{A}$ be an alphabet of cardinal q and $n \leqslant 1$. We call the repetition code of length n over $\mathscr{A}$ the code

$$C = \{(m, ..., m) \in \mathscr{A}^n; m \in \mathscr{A}\}.$$

The minimum distance of the code is n (either all the characters are identical, or all distinct). It is thus a $(n, 1, n)_q$ code. The information rate of this code is $R = \frac{1}{n}$ and its correction rate is $\tau = \left\lfloor \frac{n-1}{2} \right\rfloor$. It has therefore a rather bad information rate, but we will see in the following that it has a great correction rate.

Simple encoding and decoding algorithms for this code are natural. For encoding, we cut the message into isolated characters, and for each of these characters, we send the character n times. For decoding, after having received n characters, we decode by the character which appears the

most. If there are less than τ errors, the decoding is correct.

The last two properties allow us to understand how a correcting code can correct a transmission error and under what conditions. Property 2.3 explains that if less than τ characters were mistransmitted, we can correct the error by finding the closest word in C for the Hamming distance. Property 2.2 shows that if there were fewer than $d-1$ transmission errors, one can detect that there were transmission errors, but not necessarily correct the error correctly.

One of the goals of the theory of correcting codes is to find codes with both high information rate and high minimum distance. In addition, we ask to have efficient coding and decoding algorithms.

2.2 Linear correcting codes

In order to simplify the creation of codes, we want to use usual algebraic structures. For that, we will suppose that the alphabet $\mathscr{A}$ is $\mathbb{F}_q$ the field with q elements, and we will study a specific type of codes : linear codes.

Definition 2.6. We say that C is a $[n,k,d]_q$ **linear code** if C is a $\mathbb{F}_q$-subvector space of $\mathbb{F}_q^n$ of dimension k and minimum distance d.

C is then the image of $\mathbb{F}_q^k$ by an injective linear application with values in $\mathbb{F}_q^n$. We call **generating matrix of** C any matrix G representing such a linear injective function in the canonical bases of $\mathbb{F}_q^k$ and $\mathbb{F}_q^n$, e.g. such that $C = u \cdot G; u \in \mathbb{F}_q^k$.

Classical results of linear algebra ensure that G is a $k \times n$ matrix over $\mathbb{F}_q$ of rank k.

Remark 2.2 (Notations). We choose to note the elements of $\mathbb{F}^l$ as row vectors.

Remark 2.3. The repetition code over $\mathbb{F}_q$ seen as an example above is a $\mathbb{F}_q$-linear code. A generating matrix is $\begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$.

Definition 2.7. Let $x \in F_q^n$. We call **support of** x the set

$$Supp(x) = \{i \in [\![1,n]\!] | x_i \neq 0\}.$$

Definition 2.8. We call **Hamming weight** the function ω over $\mathbb{F}_q^n$ such that

$$\forall x \in \mathbb{F}_q^n, \omega(x) = \text{card}(\text{Supp}(x)).$$

Remark 2.4. Let $x, y \in \mathbb{F}_q^n, d(x,y) = \omega(x-y)$.

Proposition 2.4. Let C be a $[n,k,d]_q$ linear code, then

$$d = \min_{x \in C \setminus \{0\}} \omega(x).$$

Proof

Let $x, y \in \mathbb{F}_q^n$ such that $d(x,y) = d$, then $x - y \in C \setminus \{0\}$ because the code is linear, and $\omega(x-y) = d(x,y) = d$, therefore $\min_{x \in C \setminus \{0\}} \omega(x) \leqslant d$.

Let $x \in C \setminus \{0\}$ then $\omega(x) = d(x,0) \geqslant d$, because $0 \in C$. Therefore, $\min_{x \in C \setminus \{0\}} \omega(x) \geqslant d$, and

$$d = \min_{x \in C \setminus \{0\}} \omega(x).$$

Theorem 2.1 (Singleton bound)**.** *Let C be a $[n,k,d]_q$ linear code, then*

$$d \leqslant n - k + 1.$$

Proof

Let us assume for the sake of argument that all the words of C are mutually distinct on the first $k-1$ coordinates. We can then construct a bijection between $\mathbb{F}_q^{k-1}$ and C, which is absurd because $\text{card}(C) = q^k > q^{k-1} = \text{card}(\mathbb{F}_q^{k-1})$. So there are two distinct words of C which coincide on the $k-1$ first coordinates. Therefore $d \leqslant n - (k-1) = n - k + 1$.

Definition 2.9. We call **MDS (Maximal Distance Separable)**a $[n,k,n-k+1]_q$ linear code.

According to what has been seen in the first sub-section, MDS codes offer the best correction rate among linear codes of fixed length and dimension : they are therefore optimal from this point of view and potentially interesting.

Example 2.2. The repetition code of length n is MDS for all $n \geqslant 1$.

Let's make a summary of the benefits of linear codes seen so far. We have seen that linear codes are generated by matrices. This aspect is interesting because it guarantees that the objective of ease of encoding is met. Indeed, in the standard case, we have to store the whole mapping table between A^k and C and search in a table of q^k elements of length n. In the linear case, we have to store a matrix $k \times n$ and do a matrix operation in complexity $\mathcal{O}(n \times k)$. We have thus succeeded in linearizing the spatial complexity for parameter k and also the temporal complexity, this time at the cost of a multiplication by n. For the values of k and n that we want to use in practice (at least a few hundreds for k), we systematically gain a lot.

Moreover, we have given a bound on the minimum distance, which we have seen to be reached. We have thus defined a form of optimality of correction rate for linear codes. However, we still have not given a way to determine d efficiently, without enumerating the $q^k - 1$ non-zero elements of C. Moreover, an efficient way to check that a word is in the linear code would also be useful. So we will investigate a tool that can help to meet both of these expectations.

Definition 2.10. Let C be a $[n,k,d]_q$ linear code. We define the **dual code of** C as

$$C^\perp = \{u \in \mathbb{F}_q^n | \forall v \in C, < u, v >= 0\}$$

where $< ., . >$ denotes the usual Euclidean scalar product.

It follows from classical results of linear algebra that $C^\perp$ is a linear code of length n and dimension $n-k$. Let H be a generating matrix of the code $C^\perp$, we call H **control matrix of** C.

Proposition 2.5. Let C be a $[n,k,d]_q$ code, and G a generating matrix of C.

— Let $H \in \mathscr{M}_{n-k,n}(\mathbb{F}_q)$. Then H is a control matrix of C if and only if $HG^T = 0$ and $\text{rank}(H) = n - k$.

— Let H be a control matrix of C and $c \in \mathbb{F}_q^n$, then

$$c \in C \text{ if and only if } Hc^T = 0.$$

Proof

— Let $H \in \mathscr{M}_{n-k,n}(\mathbb{F}_q)$ such that $HG^T = 0$ and $\text{rank}(H) = n-k$. Then the $n-k$ rows of H form a basis of a vectorial subspace E of $\mathbb{F}_q^n$ of dimension $n-k$, the rank of H. We show that $E = C^\perp$.

Let h be a vector of $\mathbb{F}_q^n$ given by a row of H, then for all vector g in $\mathbb{F}_q^n$ given by a row of G, we have $< h, g >= hg^T = 0$ because $HG^T = 0$. Since the rows of G form a basis of C and the rows of H form a basis of E, we get from the bilinearity of the scalar product :

$$\forall u \in E, \forall v \in C, < u, v >= 0.$$

Therefore $E \subset C^\perp$, and since $\dim\ E = \dim\ C^\perp$, then $E = C^\perp$.

Conversely, let H be a control matrix of C, then $\text{rank}(H) = \dim\ C^\perp = n-k$. As H generates $C^\perp$ and G generates C, as we have seen above, we have $HG^T = 0$.

— Let $c \in \mathbb{F}_q^n$ then $c \in C$ if and only if $c \in C^{\perp\perp}$ if and only if $\forall v \in C^\perp, < v, c >= 0$ if and only if $Hc^T = 0$, because H generates $C^\perp$.

Definition 2.11. Let $x \in \mathbb{F}_q^n$, we call **syndrome of x** the quantity $S(x) = Hx^T$.

Property 2.5 showed us that the syndrome allows us to simply determine whether a word of $\mathbb{F}_q^n$ is in a code C via a matrix product, thanks to the control matrices. We can also notice that the control matrices are new descriptors of C, in addition to the generating matrices. Indeed, the control matrices describe completely the dual of C, which allows to describe completely C.It is therefore possible to define a code via a control matrix, as we will do in the next sub-section.

It remains to see the links between the dual and the minimum distance, which we formulate in this theorem :

Theorem 2.2. *Let C be a $[n,k,d]_q$ code, and H a control matrix of C. Then d is the greatest of the integers D such that all the sets of $D-1$ columns of H are linearly independent.*

Proof

We note H_i the i-th column of H for any $i \in [\![1,n]\!]$. We start by showing an equivalence. Let $p \in [\![1,n]\!]$ and $x \in \mathbb{F}_q^n$:

$$\left\{ \begin{array}{c} x \in C \\ \omega(x) = p \end{array} \right. iff \left\{ \begin{array}{c} Hx^T = 0 \\ \omega(x) = p \end{array} \right. iff \left\{ \begin{array}{c} Hx^T = 0 \\ x = (0,...,0,x_{i_1},0,..,0,x_{i_2},0,...,x_{i_p},0,...,0), x_{i_j} \neq 0 \end{array} \right.$$
$$iff \left\{ \begin{array}{c} \left(H_{i_1} | \quad ... \quad | H_{i_p}\right)(x_{i_1},...,x_{i_p})^T = 0 \\ x = (0,...,0,x_{i_1},0,..,0,x_{i_2},0,...,x_{i_p},0,...,0), x_{i_j} \neq 0 \end{array} \right. .$$

Let us note $D_{\max}$ the greatest of the integers D such that all the sets of $D-1$ columns of H linearly independent.

Let $x \in C$ such that $\omega(x) = d$, then according to the above equivalence, by selecting the columns $H_{i_1}, H_{i_2}, ..., H_{i_d}$ corresponding to the non-zero coordinates of x, we get a set of d columns

of H linearly dependent. Therefore $D_{\max} - 1 < d$, so $D_{\max} \leqslant d$.

Let us suppose for argument sake that there exists $H_{i_1}, H_{i_2}, ..., H_{i_{d-1}}$ a set of columns of H linearly dependent, then there exist $x_{i_1}, x_{i_2}, ..., x_{i_{d-1}} \in \mathbb{F}_q$ such that

$$\left(H_{i_1} | \quad ... \quad | H_{i_{d-1}}\right) (x_{i_1}, ..., x_{i_{d-1}})^T = 0.$$

According to the equivalence, we can find $x \in C$ of Hamming weight $\omega(x) = d-1$, which is absurd. Therefore any set of $d-1$ columns of H is linearly independent and $d \leqslant D_{\max}$.

Therefore $d = D_{\max}$.

It can be noticed that we have not formulated properties concerning the speed of decoding linear codes, which was one of the objectives we had formulated in the first subpart. This is because no non-exponential decoding algorithms are known in the general case. However, there are several families of linear codes for which an efficient decoding algorithm is known. We will study two of these families in the next sub-sections.

2.3 An Application : Hamming Binary Code

In this sub-section, we show that by using the properties of linear codes seen in the previous sub-section, we can easily build a family of correcting codes of correction rate 1. We also show that this family of correcting codes has a fast decoding algorithm.

We place ourselves on the alphabet $\mathbb{F}_2$. We wish to build codes of correction rate 1. We have seen that this relates to having a control matrix whose every pair of columns is linearly independent, so the columns of the control matrix must simply be distinct and non-zero. Since we know that giving a control matrix is sufficient to define a code we will proceed in this way.

Definition 2.12 ([15])**.** Let $r > 1$ be an integer. We define **the Hamming binary code $\mathscr{H}_r$ of length** $2^r - 1$ as the code given by this control matrix :

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

whose columns generate all the vectors of $\mathbb{F}_2^r$, and such that for all $j \in [\![1, 2^r - 1]\!]$, the j-th columns of H, H_j, is the binary decomposition of j (with big-endian convention).

Remark 2.5. The order of the columns does not change the constructed code. However, it allows to use a faster decoding algorithm, that we will see below.

Remark 2.6. Hamming codes are the first historical codes. They were invented to prevent errors when transmitting data for computations.

Proposition 2.6. The Hamming code $\mathscr{H}_r$ is a $[2^r - 1, 2^r - r - 1, 3]_2$ code. Its information rate is $R = 1 - \frac{r}{2^r - 1}$.

Proof

We first need to show that the code is well-defined, e.g. that H is indeed a control matrix. It is enough to show that $\text{rank}(H) = r$. It is obvious because one can extract the following submatrix :

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

which is an invertible square matrix of order r.

The length of $\mathscr{H}_r$ is indeed $2^r - 1$, therefore its dimension is $(2^r - 1) - \text{rank}(H) = 2^r - r - 1$. We calulate that $R = \frac{2^r - r - 1}{2^r - 1} = 1 - \frac{r}{2^r - 1}$.

Finally, $d \geqslant 3$ because every two columns of H are linearly independent. Since the third column is the sum of the first two, we have $d = 3$.

We will now detail the decoding algorithm. Let m be the received message. Suppose that $m = c + e$ where $c \in C$ and $\omega(e) \leqslant 1$. Then $Hm^T = Hc^T + He^T = He^T$. If $e = 0$, we detect that there is no error because the syndrome is null. If $e \neq 0$, then there exists $j \in [\![1, 2^r - 1]\!]$ such that $e = (\delta_{ij})_{i \in [\![1, 2^r - 1]\!]}$. In particular, $He^T = H_j$. So we can find j because it is the integer whose binary decomposition is given by He^T.

Remark 2.7 (Binary erasure channel). In the *binary erasure channel* model, if the number of erased characters e is less than $d - 1$, then the original word can be recovered.

Indeed, the words that can be constructed by completing what has been received are at a distance less than $d - 1$ from the word sent. According to property 2.2, there is therefore only one code word in the set of candidate words.

To decode, we proceed as follows : if Hm^T is null, we decide that $c = m$, otherwise, Hm^T is the binary decomposition of a $j \in [\![1, 2^r - 1]\!]$, and we decide that $c = m - (\delta_{ij})_{i \in [\![1, 2^r - 1]\!]} = m + (\delta_{ij})_{i \in [\![1, 2^r - 1]\!]}$. This algorithm is correct if $\omega(e) \leqslant 1$ from the previous paragraph.

2.4 Generalized Reed-Solomon Codes(GRS)

In the previous sub-section we showed that Hamming codes are relatively time efficient, especially when r increases. However, the correction rate is limited. In the following, we will try to build a family of MDS correcting codes.

In the following, we take as alphabet the $\mathbb{F}_q$ field.

Definition 2.13 ([25])**.** Let $n \in [\![1, q - 1]\!]$ and $k \in [\![1, n]\!]$. Let $\alpha_1, \alpha_2, ..., \alpha_n$ be distict non-zero elements of $\mathbb{F}_q$, and $v_1, ..., v_n$ be non-zero elements of $\mathbb{F}_q$. A **generalised Reed-Solomon code (GRS)** is given by the following control matrix : $H = V_\alpha^{(n-k)} \times \text{Diag}((v_1, ..., v_n))$ où

$$V_\alpha^{(n-k)} = \begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & ... & \alpha_n \\ \vdots & \vdots & ... & \vdots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & ... & \alpha_n^{n-k-1} \end{pmatrix} \text{ et } \text{Diag}((v_1, ..., v_n)) = \begin{pmatrix} v_1 & 0 & ... & 0 \\ 0 & v_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & v_n \end{pmatrix}.$$

We call $\alpha_1, ..., \alpha_n$ the **locators of the code**, and $v_1, ..., v_n$ the multipliers of the code.

Remark 2.8. We can point out that we are asking for $n \leqslant q-1$. Indeed, we need it to find n distinct elements in $\mathbb{F}_q^\times$. This means that if we need a correction rate τ, we must have $2\tau+1 \leqslant n-k+1 \leqslant q-k$ or $q \geqslant 2\tau+k+1$. This forces us to work in fields with many elements, which are less easy to handle.

Proposition 2.7. Un GRS code is $[n, k, n-k+1]_q$. It is therefore MDS.

Proof

We can assume that $v_1, ..., v_n$ are equal to 1, since multiplying by an invertible coefficient does not change the dimension or the minimum distance. We note that any set of $n-k$ columns of H forms a Vandermonde matrix of distinct locators. In particular, this matrix is invertible. So H has rank $n-k$ and $d \geqslant n-k+1$. So the code is of dimension k and $d = n-k+1$.

Theorem 2.3. *The dual code of a GRS code is a GRS code with same locators.*

Proof

Let C be a GRS $[n, k, n-k+1]_q$ code whose locators are $\alpha_1, ..., \alpha_n \in F_q^\times$, and whose multipliers are $v_1, ..., v_n \in \mathbb{F}_q^\times$. Thus, $H = V_\alpha^{(n-k)} \times \text{Diag}((v_1, ..., v_n))$ is a control matrix of C. We show that there exists $G = V_\alpha^{(k)} \times \text{Diag}((v'_1, ..., v'_n))$ a generating matrix of C where $v'_1, ..., v'_n \in \mathbb{F}_q^\times$.

We make a quick analysis : we look for G of the previous form such that $GH^T = 0$ and G of rank k. We notice that a matrix G of this form is always of rank k.

Let $i \in [\![1, k]\!]$ and $j \in [\![1, n-k]\!]$. The coefficient in row i column j of GH^T is :

$$\sum_{l=1}^{n} \alpha_l^{i+j-2} v_l v'_l.$$

In other words, $GH^T = 0$ if and only if for all $m \in [\![1, n-1]\!]$, $\sum_{l=1}^{n} \alpha_l^{m-1} v_l v'_l = 0$, if and only if

$$V_\alpha^{(n-1)} \times \text{Diag}((v_1, ..., v_n)) \times \begin{pmatrix} v'_1 \\ \vdots \\ v'_n \end{pmatrix} = 0.$$

Now $\text{Diag}((v_1, ..., v_n))$ is invertible and $V_\alpha^{(n-1)}$ is a $n-1 \times n$ matrix, it has therefore a non-zero kernel. So there exists $(v'_1, ..., v'_n) \in \mathbb{F}_q^n$ non-zero verifying the above equality. It remains to show that all the components of this vector are non-zero.But if a component is null (let us say, without loss of generality after a permutation of the columns of $V_\alpha^{(n-1)} \times \text{Diag}((v_1, ..., v_n))$, the n-th), then

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & ... & \alpha_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{n-2} & \alpha_2^{n-2} & ... & \alpha_{n-1}^{n-2} \end{pmatrix} \times \begin{pmatrix} v_1 & 0 & ... & 0 \\ 0 & v_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & v_{n-1} \end{pmatrix} \begin{pmatrix} v'_1 \\ \vdots \\ v'_{n-1} \end{pmatrix} = 0.$$

But the matrix of this equality is invertible, so $(v'_1, ..., v'_{n-1})$ is zero, so $(v'_1, ..., v'_n)$ is also zero, which is contradictory with the choice of $(v'_1, ..., v'_n)$. So all components are non-zero.

Remark 2.9 (Polynomial point of view). Let C be a GRS $[n, k, n-k+1]_q$ code. We have seen that its dual was GRS as well, therefore it is generated by a matrix G. Let us suppose for now that G has the following form : $G = V_\alpha^k$. Let $c = \begin{pmatrix} c_0 & ... & c_{k-1} \end{pmatrix} \in \mathbb{F}_q^k$. We can associate this vector with the polynomial $P_c = \sum_{i=0}^{k-1} c_i X^i$. The corresponding word of C is then

$$cG = \begin{pmatrix} P_c(\alpha_1) & ... & P_c(\alpha_n) \end{pmatrix}$$

. Decoding a word of C (well transmitted) can therefore be reduced to doing a Lagrange interpolation.

Remark 2.10. Like the Hamming codes, the GRS codes have a fast decoding algorithm, called algorithm of **Welch-Berlekamp**, which we do not develop here.

The GRS codes were very much used for storage devices like CDs, DVDs or hard disks.

3 Secret Sharing Schemes (SSSs)

In this section, we present secret sharing schemes (SSSs), a field of study in cryptography. SSSs were introduced simultaneously by A. Shamir [26] and G. R. Blakley [2]. The problem that SSSs seek to solve is that of sharing a secret with a set of participants while controlling the way in which the participants can access this secret. This has practical applications, such as storing a secret key in a secure manner to guard against technical failures, or controlling access to information (military and governmental applications, or in any hierarchical system). The following examples allow to better understand what we are trying to do with SSSs.

3.1 Introductary Examples

In this section, we will often have to use randomness. By convention, we will note with a capital letter a random variable and with a lower case the evaluation of this random variable.

We will use the following lemma throughout the subsection :

Lemma 3.1. Let Y be a random variable taking values in $\mathbb{F}_q$, and U an uniform random variable over $\mathbb{F}_q$, independent from Y. Then $Y + U$ has a uniform distribution over $\mathbb{F}_q$, and is independent from Y.

Example 3.1 (Five thieves)**.** We name five thieves $p_1, ..., p_5$. These thieves have a loot that they wish to protect. However, they don't trust each other and therefore want the protection to be lifted only if all the thieves are present.

One way to do this is to use a locked door. The thieves use five locks, each with a different key, and each robber gets one of the keys.

Another way is to use a smart lock. It draws independently five elements at random, with uniform distribution over a field $\mathbb{F}_q$ (of large cardinal). We note $s_1, .., s_5$ these elements. The lock then secretly sends for every $i \in [\![1, 5]\!]$ s_i to p_i. The lock will unlock only if it receives $s := \sum_{i=1}^{5} s_i$. We can verify that the random variable S is independent of any proper subfamily of $(S_i)_{i\in[\![1,5]\!]}$, but not from $(S_i)_{i\in[\![1,5]\!]}$. Thus, the knowledge of some (but not all) values s_i does not give any information about s, so a coalition of thieves can not find s (provided that $\mathbb{F}_q$ is large enough).

We have given here two ways to protect a secret (the loot) by dividing the information among a number of participants. This is one of the interest of the SSSs that we are trying to explore.

We can see that the two ways of proceeding have some common points : same number of keys, the need to gather the whole group to open the door. However, there is a fundamental difference : knowing some of the s_i values does not decrease the security of the loot, while having a key decreases the number of locks to be open. The second method therefore provides more security.

Remark 3.1. We can model the method of locks mathematically. Let S be any random variable with values in $\mathbb{F}_q$, and $C_1, ..., C_5$ uniform random variables on $\mathbb{F}_q$ mutually independent of S and each other. The variables C_i serve as locks. We publicly display $s + \sum_{i=1}^{5} c_i$, which represents the locked door. Each thief p_k for $k \in [\![1, 5]\!]$ receives c_k, which models the key of the lock C_k. We get the same operation as with keys, while removing the loss of security that comes with owning keys.

In this first example, we have considered a simple case, where all the thieves must be together to recover the loot. But what happens if a robber loses his key, or is no longer able to participate ? The loot is then unrecoverable. To avoid this kind of problem, the rules for accessing the loot must be made more flexible, while keeping them strict enough to prevent betrayal.

Example 3.2 (n knowledgeable thieves)**.** Let n be a natural integer, and $p_1, ..., p_n$ be thieves. The thieves want their loot to be recovered only if at least k thieves are gathered, where $1 \leqslant k \leqslant n$.

A solution using locks and keys is more difficult to implement. Indeed, we use $\binom{n}{n-(k-1)}$ locks. Each lock corresponds to a subset $A \subset \{p_1, ..., p_n\}$ of cardinal $n-(k-1)$ and each participant in A receives a key of this lock. Thus, if a group of k thieves meets, for any lock and associated subset $A \subset \{p_1, ..., p_n\}$, there are at most $k-1$ of these thieves in A, so a member of the group must have the key. On the contrary, if a group of less than $k-1$ thieves meets, let us note it B, there exists a lock of associated subset A such that $B \subset \{p_1, ..., p_n\} \setminus A$, and thus the group cannot open this lock.

In this second example, we see that if we want $k = \lfloor \frac{2n}{3} \rfloor$ then the number of locks is $\binom{n}{n+1-\lfloor \frac{2n}{3} \rfloor} = \binom{n}{\lceil \frac{n}{3} \rceil + 1}$ and the number of keys is $(\lceil \frac{n}{3} \rceil + 1)\binom{n}{\lceil \frac{n}{3} \rceil + 1}$. This number grows rapidly with n, and we can see that this scheme is inefficient. Another goal of the theory of SSSs is to find schemes that reduce the number of keys of each participant.

In a last example, we will see that we can deal with cases where all participants do not have the same level of authorization.

Example 3.3 (Hierarchy, from [30])**.** In a company, there are two types of employees : managers and ordinary employees. The company's rules stipulate that in order to access certain secure files, three employees must cooperate, except in the case where two managers would like to access the file. So there is an aspect of hierarchy to consider. We note d the number of managers and w the number of ordinary employees. We associate to each employee a number, between 1 and w for an ordinary employee and between $w+1$ and $w+d$ for a manager.

We present a scheme that allows us to set up these conditions. Let $\mathbb{F}_q$ be a finite field such that $q \leqslant 1 + w + \frac{w(w-1)}{2} + d$. The company publicly sets $\alpha_1, ..., \alpha_w \in \mathbb{F}_q$ distinct non-zero elements, then sets $\alpha_{w+1}, ..., \alpha_{w+d}$, distinct non-zero elements, distinct from $\alpha_1, ..., \alpha_w$ and distinct from all the elements $\alpha_i \alpha_j (\alpha_i + \alpha_j)^{-1}$ for all $1 \leqslant i < j \leqslant w$. We notice that it is possible to find such elements thanks to the bound on q. Then a trusted third party (the CEO of the company for example) draws s, a, b secretly following a uniform distribution. Finally, the latter sends to every manager i his secret share $s_i := s + \alpha_i a$, and to every employee j his secret share $s_j := s + \alpha_j a + \alpha_j^2 b$. Finally, the secured files are encrypted using s.

We check that two managers can find the secret. We notice that if i, j are the numbers of two directors, then

$$\begin{pmatrix} s_i & s_j \end{pmatrix} = \begin{pmatrix} s & a \end{pmatrix} G$$

where $G = \begin{pmatrix} 1 & 1 \\ \alpha_i & \alpha_j \end{pmatrix}$ is an invertible Vandermonde matrix because $\alpha_i \neq \alpha_j$. So the two managers can find the secret by inverting G (which is public) and computing $\begin{pmatrix} s_i & s_j \end{pmatrix} G^{-1}$. The case of three ordinary employees works similarly.

We finally check that it is possible to find the secret s if a manager and two ordinary employees participate. If i, j are the numbers of two ordinary employees and k is the number of a manager, then their shares are given by :

$$\begin{pmatrix} s_i & s_j & s_k \end{pmatrix} = \begin{pmatrix} s & a & b \end{pmatrix} G$$

where $G = \begin{pmatrix} 1 & 1 & 1 \\ \alpha_i & \alpha_j & \alpha_k \\ \alpha_i^2 & \alpha_j^2 & 0 \end{pmatrix}$. The determinant of G is $det(G) = (\alpha_k(\alpha_i + \alpha_j) - \alpha_i \alpha_j)(\alpha_i - \alpha_j) \neq 0$ according to the assumptions of the construction of $(\alpha_l)_{l \in [\![1, w+d]\!]}$. So G is indeed invertible, and the group $\{i, j, k\}$ can find the secret.

It remains to show that a group that does not contain three employees or two managers does not find information about the secret. Let i, j be two ordinary employees, then their shares are calculated as

$$s\begin{pmatrix}1 & 1\end{pmatrix} + \begin{pmatrix}a & b\end{pmatrix}\begin{pmatrix}\alpha_i & \alpha_j \\ \alpha_i^2 & \alpha_j^2\end{pmatrix} = s\begin{pmatrix}1 & 1\end{pmatrix} + \begin{pmatrix}a & b\end{pmatrix}\begin{pmatrix}1 & 1 \\ \alpha_i & \alpha_j\end{pmatrix}\begin{pmatrix}\alpha_i & 0 \\ 0 & \alpha_j\end{pmatrix}$$

which is the same as knowing

$$s\begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}\alpha_i & 0 \\ 0 & \alpha_j\end{pmatrix}^{-1}\begin{pmatrix}1 & 1 \\ \alpha_i & \alpha_j\end{pmatrix}^{-1} + \begin{pmatrix}a & b\end{pmatrix}.$$

According to the lemma, $\{i, j\}$ cannot find any information about the secret s. We proceed in the same way for the other groups not having the permission to see the secret.

We see that in this case, we have built an SSS whose secret parts are the same size as the secret itself (e.g., the participants have one private key each), while implementing the required complex access system.

To conclude this introductory subsection, we return to the essential aspects of SSSs that we have highlighted in the previous examples, and we introduce the technical vocabulary to facilitate the development of intuition in the following. Beforehand, one must determine a **set of participants** (the thieves, the employees), as well as a **access structure** (the set of groups that can retrieve the secret, and the set of those that cannot retrieve anything). A **trusted third party, or dealer** (smart lock, CEO) draws a **secret** randomly, and follows a sharing protocol to calculate and distribute a **share** to each participant. This distribution may involve the **broadcast** of instance-specific and public elements (e.g., the α_i in the company example, which may change from one application of the protocol to another). Finally, the participants know a recovery protocol to retrieve the secret correctly in a group if that group is allowed or **qualified**. According to the Kerckhoffs principle, only the randomly drawn secret and the shares are assumed to be secret.

3.2 Access Structures and Security Model

Before defining the SSSs formally, we need to define some notions beforehand. First of all, we give ourselves an integer n, a set $P = 1, ..., n$ of participants and p_0 that we call dealer.

We note in this subsection $Q = \{p_0\} \cup P$.

Definition 3.1. We call a pair (Γ, Δ) where $\Gamma, \Delta \subset 2^P$ are disjoint sets of subsets of P an **access structure over** P. The elements of Γ are called **qualified groups** and those of Δ are called **forbidden groups**. This structure is said to be **monotonic** if Γ is increasing and Δ is decreasing, e.g. :

— $\forall A \in \Gamma, \forall A' \in 2^P$, si $A \subset A'$ alors $A' \in \Gamma$.

— $\forall B \in \Delta, \forall B' \in 2^P$, si $B' \subset B$ alors $B' \in \Delta$.

In the following, we will always assume that an access structure is monotonic. We will also suppose that $\Delta \neq \emptyset$ and $\emptyset \notin \Gamma$, we say that the access structure is supposed **non-trivial**.

Let $[\Gamma^-]$ be the set of minimal elements of Γ (e.g. such that there are no elements of Γ that they strictly contain), and $[\Delta^+]$ the set of maximal elements of Δ.

Remark 3.2. The monotonicity assumption of the access structure matches the intuition one can have : if a and b can find the secret then so can a, b and c. Conversely, if a and b cannot find the

secret together, they cannot find it separately. We see that in the examples, we made recovery or privacy proofs on the elements of $[\Gamma^-]$ and $[\Delta^+]$ exclusively.

Definition 3.2. We say that an access structure (Γ, Δ) over P is **complete** if $\Delta = \Gamma^c$. In that case, we denote the access structure by Γ only.

Remark 3.3. In the first subsection, it has always been considered that either a group could retrieve the secret or it had no access to any information about it. This corresponds to security constraints that we prefer to impose in practice. However, it is sometimes interesting to study what happens in theory if we lift this rule.

Definition 3.3. Let (Γ, Δ) be an access structure over P. We call **core of** Γ

$$\text{core}\,\Gamma = \bigcup_{A \in [\Gamma^-]} A.$$

These are the participants who effectively take part in the access structure. We say that the structure is **connected** if $P = \text{core}\,\Gamma$.

Definition 3.4. Let (Γ, Δ) be an access structure over P.We define the **dual access structure** $(\Gamma^\perp, \Delta^\perp)$:

— $\Gamma^\perp = \{A \in 2^P | A^c \in \Delta\}$.

— $\Delta^\perp = \{B \in 2^P | B^c \in \Gamma\}$.

Proposition 3.1. The dual access structure is an access structure, and $(\Gamma^\perp)^\perp = \Gamma$ and $(\Delta^\perp)^\perp = \Delta$.

Proof

$A \in (\Gamma^\perp)^\perp$ if and only if $A^c \in \Delta^\perp$ if and only if $(A^c)^c \in \Gamma$ e.g. $A \in \Gamma$. We proceed similarly for $(\Delta^\perp)^\perp$.

We give two families of well-known access structures :

Definition 3.5 (Threshold access structure)**.** Let $\mathscr{A} = (\Gamma, \Delta)$ be an access structure over P and $1 \leqslant k \leqslant n$ an integer. We say that $\mathscr{A}$ is a (k, n)-**threshold access structure** if $\Gamma = \{A \in 2^P | \text{card}(A) \geqslant k\}$ and $\Delta = \{B \in 2^P | \text{card}(B) \leqslant k-1\}$. This is a complete structure.

Definition 3.6 (Ramp access structure)**.** Let $\mathscr{A} = (\Gamma, \Delta)$ be an access structure over P and $1 \leqslant t < k \leqslant n$ an integer. We say that $\mathscr{A}$ is a (t, k, n)-**ramp structure** if $\Gamma = \{A \in 2^P | \text{card}(A) \geqslant k\}$ and $\Delta = \{B \in 2^P | \text{card}(B) \leqslant t\}$. Note that a (k, n)-threshold access structure is a $(k-1, k, n)$-ramp access structure.

Remark 3.4. We saw the threshold access structures in example 3.2. We will see later that there are much better SSSs for these structures.

In the context of cryptography, we must also propose a model containing our security assumptions and a model of the behavior of a possible adversary.

In the literature [14], adversaries are often modelized by a set of $\Delta_A \subset \Delta$, called **adversary structure**, which represents the groups of participants who might not follow the protocol correctly. These participants are called **corrupted**. We represent the passive adversaries, e.g. who follow the protocol but may seek to collude, by the set Δ, which is then called **privacy structure**. These participants are called **curious**. We can suppose that the adversary is **static**, e.g. that it chooses a set $A \in \Delta_A$ and a set $B \in \Delta$ such that $A \cup B \in \Delta$ before the start of the protocol, or that he is **adaptive**, e.g. that it can choose to actively or passively corrupt a participant at each step of the sharing protocol, as long as the set A of corrupted participants verifies $A \in \Delta_A$ and the set B of curious participants verifies $A \cup B \in \Delta$. This way of modeling possible opponents is called **mixed opponent model**.

In the following, it will be assumed that every pair of participants has a secure channel for communication. We will also assume that there is such a secure channel between the dealer p_0 and each participant (we use the *secure channel* communication model [1][6]). Finally, we suppose that $\Delta_A = \emptyset$, e.g. all the participants follow the protocol.

3.3 Matroids and SSSs

In this subsection, a definition of SSSs, called the **entropic definition**, and their relationship to matroids, in particular entropic matroids, is presented.

We use the notations from subsection 3.2 for the beginning of the subsection.

Definition 3.7. A **secret sharing scheme (SSS)** Σ over Q is a collection $(S_i)_{i \in Q}$ of discrete random variables such that S_{p_0} is non-constant. For $A \subset Q$, let S_A be the random vector $(S_i)_{i \in A}$.

Remark 3.5. The secret is represented by the variable S_{p_0} and the shares are represented by $(S_i)_{i \in P}$.

We define the function $h : 2^Q \longrightarrow \mathbb{R}$ such that $\forall A \subset Q, h(A) = H(S_A)$ where H denotes the entropy. Let us define $h(A|B) = H(S_A|S_B)$ for $A, B \subset Q$. We will allow ourselves to associate the singletons and the elements of Q as parameters of h.

Definition 3.8. We define the access structure of a SSS Σ as (Γ, Δ) where :

— $\Gamma = \{A \subset P | h(p_0|A) = 0\}$.

— $\Delta = \{B \subset P | h(p_0|B) = h(\{p_0\})\}$.

Remark 3.6. We easily check that Γ is increasing, Δ is decreasing and $\Gamma \cap \Delta = \emptyset$. Therefore, it is indeed an access structure.

Remark 3.7. We can see that the conditional entropy, which we introduced as the quantity of information we have about a random variable knowing another one, allows us to formalize the notion of "being able to find the secret" and "not being able to find the secret".

Remark 3.8. Note that in reality, it is the joint distribution of the random vector that interests us in an SSS to define the access structure, and not the marginal laws. We can find in the literature a definition using the joint distribution directly ([30]). It is equivalent. In the same way, one could have defined the access structure in an equivalent way using the notions of independance and conditional probability, according to what has been stated in property 1.2.

Definition 3.9. An SSS is said to be **perfect** if its access structure is complete.

Example 3.4. Let $n = 3$. Let $S = (S_\alpha, S_\beta)$ and $U = (U_\alpha, U_\beta)$ be two random variables of uniform distribution on $\mathbb{F}_q^2$ where $\mathbb{F}_q$ is a finite field. We suppose in addition that $S_\alpha, S_\beta, U_\alpha, U_\beta$ are mutually independent. We define an SSS Σ on Q by :

— $S_{p_0} = S$.
— $S_1 = S + U$.
— $S_2 = U_\alpha$.
— $S_3 = U_\beta$.

We calculate that :

— $h(p_0) = 2\log(q)$ because S_{p_0} has a uniform distribution on $\mathbb{F}_q^2$.
— $h(p_0|i) = 2\log(q), \forall i \in [\![1,3]\!]$, because S is independent from $S_i, \forall i \in [\![1,3]\!]$.
— $h(p_0|\{1,2\}) = h(p_0|\{1,3\}) = \log(q)$ because one component of S is function of known variables and the other one is independent from known variables.
— $h(p_0|\{2,3\}) = 2\log(q)$ because S is independent from U and therefore from (S_1, S_2).
— $h(p_0|P) = 0$.

Therefore the access structure of Σ is (Γ, Δ) where $\Delta = \{\emptyset, \{1\}, \{2\}, \{3\}, \{2,3\}\}$ and $\Gamma = \{\{1,2,3\}\}$.

Example 3.5 (An SSS for every complete access structure)**.** In the introductory example 3.2, an SSS is presented for any threshold access structure. We can extend the principle to any complete access structure. The idea is to forbid access to the secret to all the groups of Δ, by creating a specific "lock" for each of these groups, and by distributing the "key" to all the other participants. It is sufficient to consider only groups in $[\Delta^+]$.

Let (Γ, Δ) be an access structure. For any $B \in [\Delta^+]$, we define U_B independently a uniform random variable with values in a finite field $\mathbb{F}_q$. We define S a uniform random variable on $\mathbb{F}_q$, independent of $(U_B)_{B\in[\Delta^+]}$. We pose :

— $S_{p_0} = S$.
— For all $i \in P$, $S_i = (S + \sum_{B\in[\Delta^+]} U_B, (U_A)_{A\in\{B\in[\Delta^+]|i\in B^c\}})$.

If $B \in \Delta$, there exists $B' \in [\Delta^+]$ such that $B \subset B'$. $U_{B'}$ is independent from $(S_i)_{i\in B}$, therefore $h(p_0|B) = \log(q) = h(p_0)$ after the introductory lemma. If $A \in \Gamma$, then for all $B \in [\Delta^+]$, there exists $i \in A \setminus B$. Therefore $h(p_0|A) \leqslant H(S|S + \sum_{B\in[\Delta^+]} U_B, (U_B)_{B\in[\Delta^+]}) = 0$. Therefore, the access structure of this SSS is indeed (Γ, Δ).

Note that this SSS, like the second introductory example, is generally impractical.

Example 3.6 (An SSS for every access structure)**.** Let (Γ, Δ) be an access structure, not necessarily complete. We can create two complete access structures resulting from (Γ, Δ) : $\alpha = (\Gamma, \Gamma^c)$ and $\beta = (\Delta^c, \Delta)$. We can create an SSS $(S_i^\alpha)_{i\in Q}$ of access structure α and another $(S_i^\beta)_{i\in Q}$ of

access structure β, which are mutually independent. By convention, we will assume that $S_{p_0}^{\alpha}$ and $S_{p_0}^{\beta}$ have an uniform distribution with values in the body $\mathbb{F}_q$.

The SSS $(S_i)_{i\in Q} = (S_i^{\alpha}, S_i^{\beta})_{i\in Q}$ has (Γ, Δ) for an access structure. Indeed, if $A \in \Gamma$ then $A \in \Delta^c$, therefore we can retrieve the two components of the secret. If $B \in \Delta$, then $B \in \Gamma^c$, therefore $H(S_{p_0}|S_B) = 2\log(q) = H(S_{p_0})$. Finally, if $C \in 2^P \setminus (\Gamma \cup \Delta)$, the C group may retrieve $S_{p_0}^{\beta}$ but not $S_{p_0}^{\alpha}$ therefore $H(S_{p_0}|S_C) = \log(q)$.

Secret sharing schemes are defined by entropy, in particular because of the links between random independence and access structures. In general, the notion of independence has an important role in this field. There are mathematical objects defined to modelize the notion of independence : the matroids, introduced in [32]. We present here some generalities about these objects. This paragraph takes up the work of C. Pabló [24], and uses the results of [8], [23] and [31].

We now abandon the notations of subsection 3.2.

Definition 3.10. We call **polymatroid** a pair (Q, f) where Q is a finite set, called **ground set** of the polymatroid, and $f : 2^Q \longrightarrow \mathbb{R}$, called **rank function**, such that :

— $f(\emptyset) = 0$.

— f is increasing : if $A \subset B$, then $f(A) \leqslant f(B)$.

— f is submodular : $f(A \cup B) + f(A \cap B) \leqslant f(A) + f(B)$.

Definition 3.11. A **matroid** M is an integer polymatroid (Q, r) (e.g. r has values in $\mathbb{N}$) such that $r(A) \leqslant |A|$ for any $A \subset Q$.

We say that a subset A of Q is **independent** if $r(A) = |A|$, otherwise we say that it is **dependent**. We call **base** a maximal independent subset for inclusion, and **circuit** a minimal dependent subset for inclusion.

Example 3.7. Let $\mathbb{F}$ be a finite field and let E be a $\mathbb{F}$-vector space. E is of finite cardinal, so we can define a matroid (E, r) where r gives the rank of a vector family (e.g. the rank of the generated vector space). One can notice that the definitions of dependent family and bases from linear algebra coincide with the definition from matroid theory.

We give characterizations of matroids by independent subsets, bases and circuits.

Proposition 3.2 (Characterization by independent subsets)**.** Let Q be a finite set, and $\mathscr{I} \subset 2^Q$. Then $\mathscr{I}$ is the set of independent subsets for a matroid of ground set Q if and only if :

— $\emptyset \in \mathscr{I}$.

— If $F \in \mathscr{I}$ and $F' \subset F$, then $F' \in \mathscr{I}$.

— If $F_1, F_2 \in \mathscr{I}$ and $|F_1| < |F_2|$, then there exists $x \in F_2 \setminus F_1$ such that $F_1 \cup \{x\} \in \mathscr{I}$. Moreover, such a matroid is unique.

Proposition 3.3 (Characterization by bases)**.** Let Q be a finite set and $\mathscr{B} \subset 2^Q$, then $\mathscr{B}$ is the set of bases for a matroid of ground set Q if and only if :

For all $B_A, b_2 \in \mathscr{B}$ and $x \in B_1 \setminus B_2$, there exists $y \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x\}) \cup y \in \mathscr{B}$.

Moreover, such a matroid is unique.

Proposition 3.4 (Characterization by circuits)**.** Let Q be a finite set and $\mathscr{C} \subset 2^Q$, then $\mathscr{C}$ is the set of circuits for a matroid of ground set Q if and only if :

— $\emptyset \notin \mathscr{C}$.

— $\mathscr{C}$ is an antichain, e.g. $C_1 \not\subset C_2$ si $C_1, C_2 \in \mathscr{C}$ and $C_1 \neq C_2$.

— If $C_1, C_2\mathscr{C}$ are distinct and $x \in C_1 \cap C_2$, then there exists $C_3 \in \mathscr{C}$ such that $C_3 \subset (C_1 \cup C_2) \setminus \{x\}$.

Moreover, such a matroid is unique.

Now, we define some operators on polymatroids.

Definition 3.12. Let $\mathscr{S}_1 = (Q, f_1)$ and $\mathscr{S}_2 = (Q, f_2)$ be two polymatroids with same ground set Q, and $c \in \mathbb{R}_+^*$. We define :

— $\mathscr{S}_1 + \mathscr{S}_2 = (Q, f_1 + f_2)$. We call $\mathscr{S}_1 + \mathscr{S}_2$ the **sum** of the polymatroids $\mathscr{S}_1$ and $\mathscr{S}_2$, it is a polymatroid.

— $c\mathscr{S}_1 = (Q, cf_1)$. We call $c\mathscr{S}_1$ a **multiple** of the polymatroid $\mathscr{S}_1$, it is a polymatroid.

Definition 3.13. Let M be a matroid of ground set Q and bases set $\mathscr{B}$. We define M^* the **dual matroid of** M as the matroid whose bases set is :

$$\mathscr{B}^* = \{B \subset Q | Q \setminus B \in \mathscr{B}\}.$$

Proposition 3.5. Let $M = (Q, r)$ be a matroid. We have the following properties :

— M^* is a matroid.

— $(M^*)^* = M$.

— The rank function of M^*, that we note r^*, is given by :

$$\forall A \subset Q, r^*(A) = |A| - r(Q) + r(Q \setminus A).$$

We can see that we find many known properties of rank functions of vector spaces. Matroids generalize the notion of linear independence. We define hereafter the family of matroids based on the linear independence relation.

Let $\mathbb{K}$ be a field, not necessarily finite.

Definition 3.14. Let (Q, f) be a polymatroid. We say that it is **$\mathbb{K}$-linearly representable, or $\mathbb{K}$-linear** if there exists $(V_i)_{i \in Q}$ a family of vector spaces such that

$$\forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i.$$

Proposition 3.6. Let (Q, f) be a $\mathbb{K}$-linear polymatroid. It is a matroid if and only if $\dim V_i \leqslant 1, \forall i \in Q$.

Remark 3.9. It is equivalent to define linear matroids using vectors as elements of the foundation Q.

With these definitions, we immediately see the connexion between linear matroids and the notions of linear independence of vectors in a vector space.

Another well known form of independence is the independence of random variables. We see that this notion allows us to define a family of matroids as well.

Definition 3.15. Let (Q, h) be a polymatroid, we say that it is **entropic** if there exists $(S_i)_{i_Q}$ a family of discrete random variables such that

$$h(\emptyset) = 0 \text{ and } \forall \emptyset \neq A \subset Q, h(A) = H(S_A).$$

We say that a polymatroid (Q, f) is **polyentropic** if there exists $c \in \mathbb{R}_+^*$ and (Q, h) an entropic polymatroid such that $(Q, f) = c(Q, h)$.

The following property shows the existence of such a polymatroid for any family of discrete random variables.

Proposition 3.7. Let $(S_i)_{i \in Q}$ be a family of discrete random variables, and $h : 2^Q \longrightarrow \mathbb{R}$ such that :

— $h(\emptyset) = 0$.

— $\forall \emptyset \neq A \subset Q, h(A) = H(S_A)$.

Then h is a polymatroid of ground set Q.

Proof

We have $h(\emptyset) = 0$, and Property 1.3 ensures that h is increasing.

If $A \subset B \subset Q$ then

$$h(B) = H(S_B) = H(S_A S_{B \setminus A}) = H(S_A) + H(S_{B \setminus A} | S_A).$$

Therefore :

$$\begin{aligned} h(A \cup B) &= H(S_A) + H(S_{B \setminus A} | S_A) \\ &\leqslant H(S_A) + H(S_{B \setminus A} | S_{A \cap B}) \\ &= H(S_A) + H(S_B) - H(S_{A \cap B}) \\ &= h(A) + h(B) - h(A \cap B) \end{aligned}$$

We can notice that the definition of entropic polymatroids involves SSSs. The property shows that in fact to each SSS corresponds an entropic polymatroid. This is sometimes interesting because it allows us to use matroid theory to make independence proofs (e.g. to show that a group is forbidden).

To be able to do this, we have to define the notion of access structure of a matroid. We start by defining the notion of conditional rank of a polymatroid, by analogy with the entropy.

Definition 3.16. Let (Q, f) be a polymatroid, let $X, Y \subset Q$, we define

$$f(X|Y) = f(X \cup Y) - f(Y).$$

Proposition 3.8. The increase and submodularity properties of the rank function imply that $f(X|Y) \geqslant f(X|Y \cup Z)$ and $f(X|Y) \geqslant 0$.

Definition 3.17. Let $\mathscr{S} = (Q, f)$ be a polymatroid, and $P_0 \subset Q$ such that $f(P_0) > 0$. We pose $P = Q \setminus P_0$. We define the access structure of $\mathscr{S}$ as the pair (Γ, Δ) where :

— $\Gamma = \{A \subset P | f(P_0|A) = 0\}$.

— $\Delta = \{B \subset P | f(P_0|B) = f(P_0)\}$.

Remark 3.10. The access structure is invariant by multiplication.

When $Q = \{p_0\} \cup P$, the access structure of an entropic polymatroid when $P_0 = \{p_0\}$ coincides with the access structure of the associated SSS.

We now see a theorem that gives an example of the interest that matroids can have.

Theorem 3.1. *Every linear polymatroid over a finite field is polyentropic.*

Proof

Let $\mathbb{K}$ be a finite field, E be a $\mathbb{K}$-vector space, (Q, f) a $\mathbb{K}$-linear polymatroid and $(V_i)_{i \in Q}$ vector subspaces of E such that $\forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i$. We restrict E to $\sum_{i \in Q} V_i$ in order to assume E of finite dimension.

Let us now define, for any $i \in Q$, $W_i = V_i^{\perp} \subset E^*$. Let there be a family of vector spaces $(E_i)_{i \in Q}$ and an injective linear application $\pi : E^* \longrightarrow \prod_{i \in Q} E_i$ such that the induced applications $\pi_i : E^* \longrightarrow E_i$ are surjective, and that for any $i \in Q$, $W_i = \ker(\pi_i)$ (one can for example consider orthogonal projectors parallel to W_i). Let U be a random variable with uniform distribution on E^*. Then $(\pi_i(U))_{i \in Q}$ form a family of random variables. We note, for all $i \in Q, S_i = \pi_i(U)$, for all $A \subset Q, S_A = (S_i)_{i \in A}$ and $\forall x \in E^*, \pi_A(x) = (\pi_i(x))_{i \in A}$.

We can easily show that

$$\forall A \subset Q, H(S_A) = \operatorname{rank}(\pi_A) \log |\mathbb{K}| = (\dim\ E^* - \dim\ \ker(\pi_A)) \log |\mathbb{K}|.$$

Let (Q, h) be the entropic polymatroid associated with the S_i variables. We notice that

$$\begin{aligned} \forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i = \dim\ E - \dim \left(\sum_{i \in A} V_i \right)^{\perp} = \dim\ E - \dim \bigcap_{i \in A} W_i = \dim\ E^* - \dim\ \ker(\pi_A) \\ = \frac{H(S_A)}{\log |\mathbb{K}|} = \frac{h(A)}{\log |\mathbb{K}|}. \end{aligned}$$

Thus, (Q, f) is polyentropic.

Remark 3.11. The family of random variables constructed in the proof is important. It will be referred to in the following as the **family of $\mathbb{K}$-linear** random variables associated with the (Q, f) polymatroid.

We have isolated a family of entropic polymatroids whose access structure is that of a linear polymatroid, which is therefore expressed using linear algebra. We are therefore interested in the SSSs associated to these matroids, which we call **linear SSSs, or LSSSs**. We know mainly two constructions for LSSSs, which we study in the following.

3.4 Linear SSSs : Shamir's construction

This construction is the first historical construction, introduced by Shamir [26]. We begin by presenting the original example.

Let $\mathbb{F}_q$ be a finite field and $1 \leqslant k \leqslant n \leqslant q-1$ two integers. We find an SSS whose access structure is a (k,n)-threshold access structure.

We draw the secret $s \in \mathbb{F}_q$ randomly with a uniform distribution. We also draw $c_1, ..., c_{k-1} \in \mathbb{F}_q$ randomly with a uniform distribution. We obtain a random vector in $\mathbb{F}_q^k$ with uniform distribution, which we associate with the polynomial $\Pi = \sum_{i=1}^{k-1} c_1 X^i + s$ de $\mathbb{F}_q[X]$. We take in any way $\alpha_1, ..., \alpha_n \in \mathbb{F}_q^\times$ which are known by all participants. The share of i is then $s_i = \Pi(\alpha_i)$.

Suppose that a group of j participants colludes. If $j \geqslant k$ then by Lagrange's interpolation theorem, we can find Π from j shares (j valuations of a polynomial of degree less than k). If $j \leqslant k-1$, then for any $c_0 \in F_q$, there exists q^{k-j-1} polynomials of degree less than k passing through the j points and through c_0 at 0. Therefore $\mathbb{P}(s = c_0) = \frac{q^{k-j-1}}{q^{k-j}} = \frac{1}{q}$. So the entropy of s knowing j shares is not modified. So the access structure of this SSS is a (k,n)-threshold access structure.

We may represent the computation of shares with the following :

$$\begin{pmatrix} s & c_1 & ... & c_{k-1} \end{pmatrix} G = \begin{pmatrix} s_1 & s_2 & ... & s_n \end{pmatrix} \text{ où } G = \begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & ... & \alpha_n \\ \vdots & \vdots & ... & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & ... & \alpha_n^{k-1} \end{pmatrix}.$$

We are interested in the generalization of this method where the secret s and the c_i components are drawn at random, and the fragments are generated by the multiplication by any matrix $G \in \mathscr{M}_{k,n}(\mathbb{F}_q)$. We must assume G is one-to-one : indeed, if G is not one-to-one, several antecedent vectors (and thus several secrets potentially) can generate the same vector of fragments.

We can see G as the generating matrix of a correcting code [21]. This makes a lot of sense, since finding the vector $\begin{pmatrix} s & c_1 & ... & c_{k-1} \end{pmatrix}$ from some shares $s_{i_1}, ..., s_{i_l}$ is the same as trying to find the word of a code by knowing some components, and not knowing the other components. This is the *binary erasure channel* problem.

Remark 3.12. Shamir's SSS uses Reed-Solomon codes in particular. One can notice that in this context, the choice of alphabet bounds the number of participants. Even if it is theoretically possible to add participants after an initial distribution, it remains that this bound cannot be exceeded. The choice of q is therefore important.

We thus present the first construction of SSSs based on error correcting codes :

Definition 3.18 (Shamir's construction)**.** Let C be a linear $[n,k,d]_q$ code whose generating matrix is G, where $2 \leqslant k \leqslant n$. We define the SSS based on Shamir's construction generated by G on the set of participants $P = [\![1, n]\!]$ as :

- S_{p_0} a uniform random variable in $\mathbb{F}_q$, which is the dealer's secret.
- $C_1, ..., C_{k-1}$ are independent uniform random variables in $\mathbb{F}_q$, independent from S_{p_0}.
- $\begin{pmatrix} S_1 & ... & S_n \end{pmatrix} = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} G$ define the shares.

Remark 3.13. Shamir's historical SSS is thus a special case where C is the dual code of a GRS code.

Remark 3.14. We see that the SSS is associated with a matroid of the form seen in Theorem 3.1. Indeed, the vector $\begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix}$ has a uniform distribution on F_q^k, and

$$\begin{pmatrix} S_{p_0} & S_1 & ... & S_n \end{pmatrix} = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} (\varepsilon|G)$$

where ε is the column $\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$. We can check that $(\varepsilon|G)$ is one-to-one because G is. Then, each column is surjective or null (there are linear forms). Therefore, $(\varepsilon|G)$ can be seen as a one-to-one application from $E = \mathbb{F}_q^k$ to $\prod_{i\in[\![0,n]\!]} E_i$ where $E_i = \mathbb{F}_q$ if the associated column is null, or $E_i = \{0\}$ otherwise.

We may see the vector spaces V_i defining the linear polymatroid associated with the $\mathbb{F}_q$-linear lines generated by the columns of $(\varepsilon|G)$.

Remark 3.15. In practice, the columns are never zero : this would mean having useless shares.

In the following, we note $\varepsilon = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and $(g_i)_{1\leqslant i\leqslant n}$ the columns of G,generating a $[n,k,d]_q$ code.

The following theorems, regarding linear SSSs, are taken from [11].

Theorem 3.2 (Access structure of Shamir's construction)**.** *The access structure of the SSS based on Shamir's connstruction generated by G is complete and the set of qualified groups is :*

$$\Gamma = \{A \subset P | \exists (x_1, ..., x_n) \in \mathbb{F}_q^n, \varepsilon = \sum_{i=1}^{n} x_i g_i \text{ et } Supp((x_1, ..., x_n)) \subset A\}.$$

In other words, these are the subsets A of $[\![1,n]\!]$ such that there is a linear dependence relation between ε and the $(g_i)_{i\in A}$.

We first show this lemma :

Lemma 3.2. Let $\mathbb{K}$ be the finite field with q elements, E a $\mathbb{K}$-vector space of finite dimension and $(V_i)_{i\in Q}$ a family of vector subspaces of E defining a linear polymatroid (Q, f). Let $(S_i)_{i\in Q}$ be a SSS of $\mathbb{K}$-linear random variables constructed as done in Theorem 3.1. Then the access structure of the SSS is :

— $\Delta = \{A \subset P | V_{p_0} \cap \left(\sum_{i\in A} V_i\right) = \{0\}\}$.

— $\Gamma = \{A \subset P | V_{p_0} \subset \sum_{i\in A} V_i\}$.

Proof: lemma

We take the notations from the proof of Theorem 3.1. The access structure of the SSS is the same as that of the entropic polymatroid (Q, h). Since $h = \log(q) f$ where $\forall A \subset Q, f(A) = \dim \sum_{i\in A} V_i$, then it is also the same as that of (Q, f).

So, injecting the expression of f into the definition of the access structure of (Q, f), we get :

— $\Delta = \{B \subset P | \dim \left(V_{p_0} + \sum_{i\in B} V_i\right) = \dim V_{p_0} + \dim \sum_{i\in B} V_i\}$.

— $\Gamma = \{A \subset P | \dim\ \left(V_{p_0} + \sum_{i\in A} V_i\right) = \dim\ \sum_{i\in A} V_i\}$.

The lemma is immediately deduced.

Proof: theorem

We use the previous lemma, reminding that in this case $V_{p_0} = \text{Vect}(\varepsilon)$ and for all $i \in P$, $V_i = \text{Vect}(g_i)$. Since all these spaces are of dimension 1, the previous lemma assures us that the structure is complete. Thus, for any $A \subset P$:

$$A \in \Gamma \text{ iff } V_{p_0} \subset \sum_{i\in A} V_i \text{ iff } \text{Vect}(\varepsilon) \subset \sum_{i\in A} \text{Vect}(g_i)$$
$$\text{iff there exists } (x_i)_{i\in A} \text{ such that } \varepsilon = \sum_{i\in A} x_i g_i.$$

This ends the proof of the theorem.

The theorem gives a necessary and sufficient condition to find the secret : if there exists $x_{i_1}, ..., x_{i_l} \in \mathbb{F}_q$ such that $\sum_{j=1}^{l} x_{i_j} g_{i_j} = \varepsilon$, we can retrieve the secret by remarking that :

$$S_{p_0} = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} . \varepsilon = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} (\sum_{j=1}^{l} x_{i_j} g_{i_j})$$
$$= \sum_{i=1}^{l} x_{i_j} \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} . g_{i_j} = \sum_{j=1}^{l} x_{i_j} S_{i_j}.$$

Remark 3.16. The theorem tells us that the access structure depends on the choice of G.

In the following, we wish to study the access structures of SSSs based on the Shamir construction.We cite the results of [12], assuming that the codes are MDS.

Theorem 3.3. *Suppose that C is a MDS code. Then any subset of k elements at least is qualified.*

Theorem 3.4. *Suppose that C is a MDS code, and that there exist $i_0 \in [\![1, n]\!], a \in \mathbb{F}_q^\times$ such that $\varepsilon = a g_{i_0}$. Then a subset is qualified if and only if it contains at least k elements or it contains i_0.*

We can see that this type of SSS is useful for setting up a hierarchical system, where a manager must know the secret and we want a threshold for the other participants. We can formulate a similar theorem, where two shares are particular this time :

Theorem 3.5. *Suppose that C is a MDS code, and that ε is a linear combination of g_1 and g_2, but not of g_1 or g_2 only. Let $A = \{i_1, ..., i_l\} \subset P$, then A is qualified if and only if one of the following is true :*

- *$l \geqslant k$.*
- *$l \leqslant k-1$ and $1, 2 \in A$.*
- *$l = k-1$, rank$(\varepsilon, g_{i_1}, ..., g_{i_l}) = k-1$ and $1, 2 \notin A$.*

Again, we see that we can set up two hierarchies of shares. We have a last theorem related to the columns of G.

Theorem 3.6. *Suppose that C is a MDS code. Then the access structure of the SSS generated by G is a (k,n)-threshold access structure if and only if ε is not a linear combination of any set of $k-1$ columns of G.*

Remark 3.17. In particular, this is the case for Shamir's historical SSS using Vandermonde matrices.

It has been assumed in all the previous theorems that the codes are MDS : this is because the duals of MDS codes are MDS according to [10], which allows to control the linear independence relations of the columns of G. We see a theorem highlighting this.

Theorem 3.7. *Let G be a generating matrix of a $[n,k,d]_q$ linear code C. We note $d^\perp$ the minimum distance $C^\perp$. If $A_1 \in [\Gamma^-]$ and $A_2 \in [\Gamma^-]$ then*

$$|A_1 \cup A_2| - |A_1 \cap A_2| \geqslant d^\perp.$$

It seems that Shamir's construction is useful to build SSS with hierarchical access structures. Moreover, this construction offers the possibility to choose the secret. However, the access structures are difficult to describe, and theorems require assumptions that make them impractical. The literature is mainly oriented towards the second construction.

3.5 Linear SSSs : Massey's construction

This construction is motivated by the construction of families of linear random variables (seen in the proof of Theorem 3.1). It was developed by J.L. Massey in [19] and [20]. We keep the same notations for n, k, Q and P.

Definition 3.19. Let C be a $[n+1,k,d]_q$ code with generating matrix $G = \begin{pmatrix} g_0 & g_1 & \dots & g_n \end{pmatrix}$. Let $U = \begin{pmatrix} U_1 & \dots & U_k \end{pmatrix}$ be a uniform random variable in $\mathbb{F}_q^k$, then

$$\begin{pmatrix} S_{p_0} & S_1 & \dots & S_n \end{pmatrix} = U \cdot G.$$

We say that $(S_i)_{i\in Q}$ is the SSS based on Massey's construction generated by G.

Remark 3.18. We see from Remark 3.14 that Shamir's construction resembles Massey's in the particular case where $g_0 = \varepsilon$. We see the connection between this construction and linear polymatroids in the same way.

The access structure related to this construction is specified in the following.

Theorem 3.8. *Let $(S_i)_{i\in Q}$ be a SSS based on Massey's construction generated by G, generating a code C. Its access structure is complete, and the set of qualified groups is*

$$\Gamma = \{A \subset P | \exists (1, x_1, ..., x_n) \in C^\perp, Supp((x_1, ..., x_n)) \subset A\}.$$

Proof

The proof is almost identical to that for the Shamir construction.

Remark 3.19. The theorem gives an algorithm to retrieve the secret. If $(1, x_1, ..., x_n) \in C^\perp$ then :

$$g_0 + \sum_{i=1}^{n} x_i g_i = 0 \text{ i.e. } g_0 = \sum_{i=1}^{n} (-x_i) g_i.$$

If we find $x_{i_1}, ..., x_{i_l}$ such that $g_0 = \sum_{j=1}^{l} (-x_{i_j}) g_{i_j}$ then :

$$s = u.g_0 = \sum_{j=1}^{l} (-x_{i_j}) u.g_{i_j} = \sum_{j=1}^{l} (-x_{i_j}) s_{i_j}.$$

Remark 3.20. The theorem shows that the choice of G to generate C does not matter, contrary to Shamir's construction. We can therefore speak of a SSS based on C for the Massey construction.

We fix from here the notations C and G introduced previously.
We can see that the minimum distance of the dual code still intervenes for this construction.

Theorem 3.9 ([12]). *Let Γ be the set of qualified groups of the SSS based on the Massey construction with the code C. Let $d^\perp$ be the minimum distance of $C^\perp$.*

- *Let $A \subset P$, if card$(A) \leqslant d^\perp - 2$, then $A \notin \Gamma$.*
- *There exists $A \subset P$ such that $A \in \Gamma$ and card$(A) = d^\perp$.*

To introduce the last theorem, we need to introduce some additional notions about linear codes.

Definition 3.20. We say that a vector $u \in \mathbb{F}_q^n$ covers another vector $v \in \mathbb{F}_q^n$ and we note $v \preceq u$ if $\text{Supp}(v) \subset \text{Supp}(u)$. We note $v \prec u$ if $\text{Supp}(v) \subsetneq \text{Supp}(u)$.

Definition 3.21. Let C be a linear code.

- We say that $u \in C$ is **minimal** if it covers only its multiples among the words of C (e.g. the αu for $\alpha \in \mathbb{F}_q$).
- We say that C is **minimal** if every word of C is minimal.
- **The covering problem of a linear error correcting code** is the problem of finding the set of minimal words of C.

We can now give the last theorem concerning the access structures of SSSs based on the Massey construction.

Theorem 3.10 ([13]). *Let C be a $[n+1, k, d]_q$ code and H one of its control matrices whose columns are noted $h_0, ..., h_n$. If $C^\perp$ is minimal, then the SSS based on C for the Massey construction has q^{n-k+1} elements in $[\Gamma^-]$. Moreover :*

- *If $d = 2$, then for all $i \in [\![1, n]\!]$: either h_i is a multiple of h_0, in which case i is in every minimal qualified group, or h_i is not a multiple of h_0, in which case i is in $(q-1)q^{n-k-2}$ minimal qualified group.*
- *If $d \geqslant 3$, for all $1 \leqslant t \leqslant d-2$ then every group of t participants is included in $(q-1)q^{n-k-(t+1)}$ of the minimal qualified groups.*

4 Applications

In this section, we present some applications of secret sharing schemes. One of the main interests of SSSs and in particular Shamir's is the application to secure multiparty computation, and it is in this direction that research is most active at the moment. SSS are also used in practice for secure storage. Finally, we will see that SSSs allow us to avoid the need for a trusted third party, notably thanks to the links with secure multiparty computation, which makes it possible to use SSSs in problems such as Byzantine generals. We will detail these elements in the following.

4.1 Back to the Adversary Model

Before presenting the applications, we come back to the notion of adversary model which was introduced in part 3 but not reused afterwards, and whose importance comes into play when we think about the application of a protocol. Indeed, one objective of cryptography is to set conditions and guarantees on the security of primitives. In practice, a participant is technically free and can follow or not follow a protocol. It is thus essential to anticipate the possible actions of the participants, and to impose constraints on them to guarantee the security of the protocol. This can be done by checking the information sent by each participant for example. The adversary model is used to model the behavior of the participants, and to express clearly the assumptions on this behavior, and allows to avoid incidents such as the misuse of a protocol.

As an example, we propose an attack on the Shamir SSS, introduced in [29]. We ask a SSS to take into account (at least) two security constraints in practice : confidentiality, e.g., the groups in Δ do not recover the secret, and correctness, e.g., the groups in Γ recover the secret. The attack is on the second constraint.

Let $2 \leqslant k \leqslant n \leqslant q-1$ be integers. The dealer draws at random a secret $s \in \mathbb{F}_q$ and a polynomial $\pi \in \mathbb{F}_q[X]$ of degree less than $k-1$ and such that $\pi(0) = s$. It fixes publicly n distinct nonzero locators $\alpha_1, ..., \alpha_n \in \mathbb{F}_q$. It distributes the fragments $s_i = \pi(\alpha_i)$ to the participants. Let us now suppose that k participants try to find the secret with only one malicious individual. We can assume without loss of generality that it is $1, ..., k$ and that k is malicious. The participant k can compute by Lagrange interpolation a polynomial δ of degree less than $k-1$, such that $\delta(\alpha_i) = 0$ for i less than $k-1$, and of constant coefficient chosen equal to ϵ. Finally, k shares $\delta(\alpha_k) + s_k$. The group then finds the polynomial $\pi + \delta$ of constant coefficient $s + \epsilon$. The honest participants have an incorrect secret, and even worse, k can find the secret because he knows ϵ. The dishonest participant is thus in a position to block access to the secret, to use the secret for his own benefit, or to use the secret for extortion or blackmail (in the case where another honnest participant cannot intervene to retrieve the secret). In general, $n - k + 1$ dishonest participants are sufficient to completely compromise the effectiveness of the protocol.

We can propose alternative SSSs in the case where we assume that the adversary may not follow the protocol [7]. For the rest, we continue to assume, as before, that the adversary is passive and follows the protocol. Thus, Shamir's protocol is well secured as we have shown.

4.2 Secure multiparty computation

We first define the concept of secure multiparty computation, and then show how SSSs can be used for this application. This subsection resumes the presentation [17].

One of the first problems that can be addressed by secure multiparty computation is Yao's millionaire problem [33] : two millionaires want to know who has the largest fortune of the two, without revealing their own fortune. More generally, the goal of secure multiparty computation is to compute boolean or arithmetic functions depending on parameters provided by several participants in such a way as to find correct results for each participant while keeping the parameters secret. In general, five constraints are required to be verified :

— **Confidentiality** : no participant learns more than his entry and his result.

— **Correctness** : the result of each participant is correct.

— **Independance of the entries** : each participant chooses an entry independent of those of the other participants.

— **Guaranteed delivery**(optional) : each participant receives his result.

— **Justice** : dishonest participants cannot receive a result if honest participants receive nothing.

The protocol must act as an impermeable black box in which each participant sends its parameter independently of the others, and which returns the corresponding results. In particular, the protocol does not protect against attacks on the prior choice of its parameter, as it takes place outside the box. The protocol should just serve as a secure module.

The current standard definition [5] is the following : one defines an ideal protocol involving an incorruptible trusted third party connected to each participant by a secured channel (for the confidentiality and the guaranteed delivery). Each participant sends its parameter to the trusted third party. The trusted third party calculates the results and sends them to each participant respectively. We say that a protocol is **secure** if for any set of inputs, it sends the same results as the ideal protocol and if for any attack against this protocol, there is an attack against the ideal protocol with the same effects. We call this model the *ideal/real paradigm*. In particular, a secure protocol respects the five previous constraints. The definition of security obviously depends on the chosen adversary model.

Canetti has shown in [5] that the notion of security is modular : if a sub-protocol of a more global protocol is secure (taking it as the main protocol) then it is secure as a sub-protocol (e.g. everything happens as if the trusted third party replaced the execution of the sub-protocol). In particular, if we find secure protocols for elementary arithmetic and boolean functions, we can deduce secure protocols for all arithmetic and boolean functions.

> *Remark* 4.1. When composing the modules, the result must not be evaluated otherwise everyone will know the intermediate steps of the calculation. For example, for three parameters x_1, x_2 and x_3, the secure calculation of $x_1 + x_2 + x_3$ can be broken down as calculating $x_1 + x_2$ and then adding x_3 to the result. However, the way to securely compute $x_1 + x_2$ must not allow participants to immediately find the value of $x_1 + x_2$, otherwise the participant knowing x_2 may find x_1, and then at the end of the computation x_3, when he should not have been able to. We will see in the following a way to manipulate $x_1 + x_2$ in calculations without knowing the value of $x_1 + x_2$ by using SSSs.

We will now show how we can do secure multiparty computation using Shamir's SSS. To do this, we need to assume that among n participants, strictly more than $\frac{n}{2}$ participants are honest. We note $t = \frac{n-1}{2}$. In the case of SSS, an arithmetic secure multiparty computation proceeds in three steps :

— *Initial share* : each participant shares his secret with the others using Shamir's protocol with a polynomial of degree t π_i and $\alpha_1, ..., \alpha_n$ locators publicly predetermined and common to all participants. Note that the secret cannot be recovered by the adversary, because the threshold is $t + 1 \geqslant \frac{n}{2}$.

— *Step-by-step computation* : at each step of the secure computation, the participants compute new shares, so that the new polynomial found after the interpolation of these shares (if it was done) is of degree less than t and of constant coefficient equal to a sum or a product of secrets already obtained. The necessary calculations are detailed below.

— *Recovery* : the participants interpolate the last computed shares, and obtain the desired result.

> *Remark* 4.2. Since polynomials of degree less than t have their non-constant coefficients random, the adversary cannot predict that a polynomial is of degree strictly less than t. The assumption that more than $\frac{n}{2}$ participants are honest is therefore sufficient to guarantee confidentiality.

Remark 4.3. Note that since many participants are needed to recover the secret, all participants must receive the same result.

It remains to detail the processes to be carried out to sum or multiply secrets.

For the sum, it is assumed that the participants have shares $a(\alpha_i)$ and $b(\alpha_i)$ that allow to reconstruct a and b two polynomials of degree less than t, whose constant coefficients $s_a = a(0)$ and $s_b = b(0)$ are secret. Participant i can compute $c(\alpha_i) = a(\alpha_i) + b(\alpha_i)$ a new shares of a polynomial $c = a + b$, whose constant coefficient $c(0) = a(0) + b(0) = s_a + s_b$. Moreover, the degree of c is smaller than t.

For the product, it is again assumed that the participants have shares $a(\alpha_i)$ and $b(\alpha_i)$ allowing to reconstruct a and b two polynomials of degree less than t, whose constant coefficients are $s_a = a(0)$ and $s_b = b(0)$. The participant i can then compute $c(\alpha_i) = a(\alpha_i)b(\alpha_i)$. This time, the polynomial c is equal to ab. So it has a constant coefficient $c(0) = a(0)b(0) = s_a s_b$ but it is of degree less than $2t$. So we will have to use a degree reduction technique introduced in [9].

It is momentarily assumed that the participants can obtain the shares of two polynomials R_t and R_{2t}, of degrees t and $2t$ respectively, whose coefficients are all drawn randomly and secretly, and are mutually independent except for the relation $r := R_t(0) = R_{2t}(0)$. Participant i can then compute $d(\alpha_i) = c(\alpha_i) - R_{2t}(\alpha_i)$ the share of a polynomial d of degree $2t$ and such that $d(0) = s_a s_b - r$. The participants can then interpolate the fragments of d to find the polynomial d, since $2t + 1 \leqslant n$. Note that c remains completely secret because its coefficients are hidden by those of R_{2t}. The participant i can then compute a share $c'(\alpha_i) = R_t(\alpha_i) + d(0)$. The polynomial c' is indeed of degree less than t, and of constant coefficient $c'(0) = R_t(0) + d(0) = s_a s_b$.

It remains to explain how to calculate the shares $R_t(\alpha_i)$ and $R_{2t}(\alpha_i)$. For any $i \in [\![1, n]\!]$, the participant i randomly draws $r_i \in \mathbb{F}_q$, then randomly and independently draws the non-constant coefficients of a polynomial R_t^i of degree t such that $R_t^i(0) = r_i$, then in the same way those of a polynomial R_{2t}^i of degree $2t$ such that $R_{2t}^i(0) = r_i$. For any $j \in [\![1, n]\!]$, i sends to j $R_t^i(\alpha_j)$ and $R_{2t}^i(\alpha_j)$. Participant j can then calculate $R_t(\alpha_j) = \sum_{i=1}^n R_t^i(\alpha_j)$ and $R_{2t}(\alpha_j) = \sum_{i=1}^n R_{2t}^i(\alpha_j)$, where $R_t = \sum_{i=1}^n R_t^i$ and $R_{2t} = \sum_{i=1}^n R_{2t}^i$. In the end, each participant has shares of the desired form.

4.3 Secure storage

The first historical application of SSSs, suggested by Shamir in [26], is in key management. Suppose one uses a secret key K for cryptographic applications. The question arises as to how to store the key. Simply keeping the key unchanged on a hard drive or other backup device is problematic : these devices may be destroyed or have malfunctions, and the key may be lost. Saving the key in multiple locations is not a good solution, as it increases the chances of an attacker gaining access to the key. Saving the key without modification is dangerous in itself because an attacker could access the memory area where the key is saved, and could then simply read the K key, since it is not protected. It is not useful to encrypt K and save the encryption, because the problem is transposed to the key used to encrypt K.

The solution proposed by A. Shamir is the following : using a Shamir SSS of parameters n and k, we create n shares $(s_i)_{i \in [\![1,n]\!]}$ of K. One then records each of the fragments on independent memory areas (for example different servers). Thus, as long as less than $n - k$ shares are lost, K can always be found. On the other hand, an attacker needs to access k shares to be able to recover K. SSSs therefore reduces both the risk of key loss and theft. This technique is used in cloud services [16].

By combining this way of saving a key with secure multiparty computation, it is possible to create systems capable of using a key that does not appear anywhere in the memory, neither during computation nor during storage.

4.4 Other mentions

S. Micali [22] suggested to use SSSs to apply it to *fair cryptography*, a branch of cryptography aiming at developing cryptographic systems guaranteeing confidentiality, while allowing justice to access the

encrypted data. His proposal is to make it mandatory to share keys via a SSS with the justice system. The use of a SSS offers security guarantees to the user of the protocol, while the sharing allows the justice system to access the encrypted data if needed.

We have seen in the sub-section devoted to multiparty computation that the participants can construct a random polynomial of arbitrary degree whose coefficients are secret for each of the participants. Thus participants are able to dispense with a dealer for the distribution of shares. In general, SSSs can be used to dispense with a trusted third party (in the specific case of a calculation, this is the principle of secure multiparty calculation). We can thus find applications of SSSs for problems related to the decentralization of operations, in particular to the problem of Byzantine agreement (for example [18]) and to the blockchain technology.

Références

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. STOC'88, 1988.

[2] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. National Computer Conference*, AFIPS, page 313–317, 1979.

[3] D. Boucher. Codes Correcteurs d'Erreurs, Aspects Algébriques. Cours de M1 à l'Université de Rennes 1,2020.

[4] E.F. Brickell and D.M. Davenport. On the Classification of Ideal Secret Sharing Schemes. *J. of Cryptology*, 4 :123–134, 1991.

[5] R Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1), 2000.

[6] D. Chaum, C. Crepeau, and I. Damgard. Multi-Party Unconditionally Secure Protokols. STOC'88, 1988.

[7] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 383 –395, 1985.

[8] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[9] I. Damgård and J. Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO 2007*, pages 572–590. Springer.

[10] Philippe Delsarte. An algebraic approach to the association schemes of coding theory. *Philips Research Reports Supplement*, 1973.

[11] C. Ding. *Concise Encyclopedia of Coding Theory*, chapter Secret Sharing with Linear Codes. Chapman and Hall/CRC, 2021.

[12] C. Ding and A. Renvall. The access structure of some secret-sharing schemes. In *Information Security and Privacy*, LNCS 1172, page 67–78. Springer, 1993.

[13] C. Ding and J. Yuan. Covering and secret sharing with linear codes. In *Discrete Mathematics and Theoretical Computer Science*, pages 11–25, LNCS 2732, 2003. Springer.

[14] S. Fehr and U. Maurer. Linear VSS and Distributed Commitments Based on Secret Sharing and Pairwise Checks. In *LNCS 2442*. CRYPTO'02, 2002.

[15] R. W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29 :147–160, 1950.

[16] T. Hansen and L. Larsen. Securing and managing secrets with hashicorp vault enterprise. AWS Blog, 2018. https ://aws.amazon.com/fr/blogs/apn/securing-and-managing-secrets-with-hashicorp-vault-enterprise/.

[17] Y. Lindell. Secure multiparty computation. *Communnications of the ACM*, 64(1) :86–96, 2021.

[18] Jian Liu, Wenting Li, Ghassan O. Karame, and N. Asokan. Scalable byzantine consensus via hardware-assisted secret sharing. *IEEE Transactions on Computers*, 68(1) :139–151, Jan 2019.

[19] J.L. Massey. Minimal codewords and secret sharing. *Proc. 6th Joint Swedish-Russian Workshop on Information Theory*, pages 276–279, 1993.

[20] J.L. Massey. Some applications of coding theory in cryptography. *Cryptography, Codes and Ciphers : Cryptography and Coding IV*, pages 33–45, 1995.

[21] R.J. McEliece and D.V. Sarwate. On sharing secrets and reed-solomon codes. *Comm. ACM*, 1981.

[22] S. Micali. Fair public-key cryptosystems. In *Proc. of CRYPTO '92, Lecture Notes in Comput. Sci.*, volume 740, pages 113–138, 1993.

[23] J.G. Oxley. *Matroid Theory.* Oxford Science Publications. The Clarendon Press, Oxford University Press, New-York, 1992.

[24] C. Padró. Lecture Notes in Secret Sharing. Éléments du cours Applications of Combinatorics to Information-Theoretic Cryptography, Nanyang Technological University, Singapore, 2013.

[25] I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *SIAM Journal of Applied Math.*, 8 :300–304, 1960.

[26] A. Shamir. How to share a secret. *Communications of the ACM*, 1979.

[27] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3) :379–423, 1948.

[28] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4) :656–715, 1949.

[29] M. Tompa and H. Woll. How To Share a Secret with Cheaters. *Journal of Cryptography*, pages 133–138, 1988.

[30] M. E. van Dijk. *Secret key sharing and secret key generation.* PhD thesis, Technische Universiteit Eindhoven, 1997.

[31] D.J.A Welsh. *Matroid Theory.* Academic Press, London, 1976.

[32] H. Whitney. On the Abstract Properties of Linear Dependence. *American J. of Math.*, 57 :509–533, 1935.

[33] A. C. Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982).* IEEE, New York.