Implementability of Black-Box Sharing Secret Schemes *

Mathieu David
UGA
Grenoble, France
mathieu.david1@etu.univ-grenoble-alpes

Supervised by: Pierre Karpman

I understand what plagiarism entails and I declare that this report is my own, original work.
Name, date and signature: DAVID Mathieu, April, the 24th of 2023

1 Introduction

In modern cryptography, a Secret sharing scheme can be defined as a pair of algorithms. The first one is a non-deterministic algorithm which will be called A such that A takes as input a secret s and a parameter n. This algorithm will return n *shares* of s. The second algorithm is a deterministic reconstruction algorithm which will be called R such that, if R has a sufficient amount of information (ie. in other words enough shares as input), R can retrieve s (this is called the completeness property). Otherwise, R cannot retrieve any information on s (this is called the privacy property).

To further detail the notion of a "sufficient amount of information", we define Γ an *access structure* as a set of subsets of $\{1, ..., n\}$ (ie. $\Gamma \in \mathscr{P}(\{1, ..., n\})$) such that if we give an element of Γ to R, we provide enough informations to retrieve s. We will note Δ as $\mathscr{P}(\{1, ..., n\})\backslash\Gamma$ (ie. the set of all elements of $\mathscr{P}(\{1, ..., n\})$ which do not allow us to achieve the secret reconstruction).

Then, we talk about a (n, t) threshold secret sharing scheme if R can retrieve the secret s if and only if R has (at least) t shares of s as input. In other word the access structure for this secret-sharing scheme is $\Gamma = \{e \in \mathscr{P}(\{1, ..., n\})$ such that $card(e) > t - 1\}$ and $\Delta = \{e \in \mathscr{P}(\{1, ..., n\})$ such that $card(e) < t\}$.

A theoretic situation where such a scheme could be useful is given in [Shamir, 1979]: "Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys to the locks each scientist must carry?"

Shamir explicits in [Shamir, 1979] a well-known method to build such a scheme relying on linear algebra and polynomial interpolation. But the fact is that this scheme requires the shares being element of a finite field whose size is function of n. More generally, for any t and n, there exists a secret sharing scheme that each values of the shares are in a finite field with a constraint on the size depending of n and t. However, we would like the secret sharing scheme can be instantiated on any finite abelian group since it imposes less constraints on the domain of definition. With secret sharing schemes that exist today, it is impossible to deploy a scheme on an imposed structure (unless if the structure is the same that the one used to develop the secret sharing scheme). That is why we would like to build a secret sharing scheme that is more convenient and works in the same way no matter the structure but it is way more difficult to do. Then, a black box secret sharing scheme (BBSSS) is a secret-sharing scheme that works in the same way no matter the finite abelian group.

R. Cramer and C. Xing, explicit in [Cramer and Xing, 2019] a construction of such a BBSSS. The purpose of this internship is to work on the feasibility of the implementation of a BBSSS.

2 Current status of the work

To provide a partial implementation of a BBSSS, a documentation step is needed. All the first part of my internship was to understand important notions related to secret sharing and BBSSS. To sum up what I have done during these first weeks, I will talk about the content of these articles and which direction I will follow for the implementation part.

2.1 More details on Black-box secret sharing schemes

To make the link with monotone span program (described in the next sub-part), we need to provide clarifications. A more technical way to define BBSSS is described in [Cramer and Xing, 2019] : we consider a matrix $\boldsymbol{M}$ over a ring R and G an arbitrary finite abelian group. In addition, we define a surjective function ψ from $\mathscr{P}(\{1, ..., n\})$ (where n is the number of shares) to $\mathscr{P}(\{1, ..., h\})$ (where h is the row dimension of the matrix). Here, ψ is usefully used to group matrix columns (resp. for entries of vectors) when h is larger than n. In other words, ψ will return one or more indices of column, for each values from 1 to n. Without loss of generality, we will consider that $n = h$ meaning that ψ is the identity.

*footnotes

A way to obtain shares from a secret $s \in G$ is to build a vector $g = (s, g_2, ..., g_n)$, where $g_2, ..., g_n$ are uniformly sampled from G, and multiply this vectore with the transpose of the matrix $\boldsymbol{M}$ (ie. $\boldsymbol{S} = g\boldsymbol{M}^\intercal$).

In the rest of the report, since we considered ψ as the identity function, we will consider a matrix $\boldsymbol{M}_S$ the submatrix composed of the row of indices in $S \in \mathscr{P}(\{1, ..., n\})$. For a vector, V_S, where $S \in \mathscr{P}(\{1, ..., n\})$, can be seen as the vector V composed of the entries which have indices in S.

We also define a *reconstruction vector* for each element a of Γ such that the product $\Lambda \cdot S_a$ allows us to retrieve s. The set $\mathscr{R}$ will define the collection of reconstruction vectors. The privacy condition said that for each element $b \in \Delta$, S_b is statistically independent from s.

2.2 Definition and purpose of a monotone span program in the construction

A monotone span program is defined in [Cramer and Xing, 2019] by a tuple of four elements : a ring R, a matrix $\boldsymbol{M}$ over the ring R, the same surjective function ψ as defined in the previous subsection on BBSSS, and a target vector called μ over the ring R such that $\mu = (1, 0, ..., 0)$. In this part, we will note $\boldsymbol{M}_T$ the submatrix of $\boldsymbol{M}$ composed of the column of indexes of T (where $T \in \mathscr{P}(\{1, ..., n\})$.

There is a direct link between access structure and monotone span program since we say that a monotone span program can compute an access structure Γ if for any element S of Γ, $\mu \in im(\boldsymbol{M}_S)$. In other words, μ is generated by the submatrix of support S. In addition, for any element T of Δ, the first column of $\boldsymbol{M}_T$ must be generated by the other column of $\boldsymbol{M}_T$ (.ie it exist a non-zero vector λ such that $\boldsymbol{M}_T\lambda = 0$).

One important result of the paper [Cramer and Xing, 2019] is that, a Monontone Span Program $(R, \boldsymbol{M}, \psi, \mu)$ computing an a access structure as defined in the introduction exists if and only if a BBSSS $(R, \boldsymbol{M}, \psi, \mathscr{R})$ computing the same access structure exists. Hence, it exists a direct link between monotone span programs and Black Box secret sharing schemes. This is important for the rest of the paper [Cramer and Xing, 2019] since this construction relies on Monotone Span Program manipulation.

2.3 Construction of R.Cramer & C.Xing

Introduction

In this part, we will explicit the construction of the matrix of a monotone span program that computes an access structure over $\mathbf{Z}$ modulo any prime number. The way of thinking here is to know that a monotone span program follows the local-global principle (also called the Hasse principle). This property says that a phenomenon is true globally if and only if it is true locally everywhere. Hence, the idea from [Cramer and Xing, 2019] is to *glue* two monotone span programs that computes such an access structure modulo any prime number $p \leq n$ and another one computing the access structure modulo prime numbers $p > n$. In the second sub section, we will shortly explained how [Cramer and Xing, 2019] create the matrix of the first monotone span program and explain more each step in the next sub sections. Indeed, in the third one, we will introduce the concept of linear code which is useful to create secret sharing scheme and we will see in the fourth sub section that we need to create a matrix modulo p^l $l \geq 1$ to achieve the conditions needed. Finally, we will conclude.

Overview of the process of creation of the first monotone span program

To sum up the idea in [Cramer and Xing, 2019] to compute the first monotone span program, we need first to compute a matrix defined on $\mathbb{Z}^{n*t}$ that generate a linear code (notion that we will detail further in the next paragraph) related to Shamir's construction in [Shamir, 1979] (where n and t are arbitrary dimensions). Secondly, we will transform the matrix using a mapping to obtain another greater matrix defined on $\mathbb{Z}^{mn*mt}$ (where m is the prime power of p) with coefficients value less or equal to $p - 1$. Thirdly, we will repeat this process for each prime number p and finally lift all these matrices to one matrix which is the desired matrix. These steps are more detailed in the next paragraph.

Linear codes

In order to create the following secret sharing schemes and manipulate them, we will need to introduce the notion of linear codes. Linear codes are a common way to build linear secret-sharing schemes. Here is the definition from [Cramer and Xing, 2019] and [Karpman, 2022]: linear codes of size n, and dimension k over $\mathbb{F}_q$, noted $[n, k]q - ary$ linear code, are k-dimensional subspace of $\mathbb{F}_q^n$ generated by a matrix $\boldsymbol{M}$. The way of building a secret sharing scheme from a linear code is quite the same as the method for monotone span program. We will not define linear codes deeper since it is not the most important notion and linear codes are related to monotone span programs.

Code in field extension of size p^l and chinese remainder theorem

As saw above, since it is easy to find a monotone span program that computes an access structure modulo a given prime number p, we generate one matrix computing the access structure modulo each prime p. However, we will need to compute a generator matrix modulo p^l $l \geq 1$ before. The reason is the fact that we need to resize the matrix if we reduce the size of the field. Indeed, if we reduce the size of the field, some coefficient of the matrix should not be represented anymore and we will lose informations about the code. The idea is to link linear code over field extensions of size p^l for $l \geq 1$ to a field of size p. Here are some idea of the properties written in [Cramer and Xing, 2019] that will show the equivalence between a code over a field of size p^l where $l \geq 1$ and p a prime number, and a code over field of size p. The paper shows that for each prime number p there exists a mapping from $\mathbb{F}_{p^m}$ to $\mathbb{F}_p^m$ ϕ such that ϕ map a

p^m-ary $[n, k]$ linear code to p-ary $[nm, km]$ linear code. The matrix generating the p-ary $[nm, km]$ ($\boldsymbol{M_1}$) linear code is simply the matrix generating p^m-ary $[n, k]$ ($\boldsymbol{M_2}$) linear code with ϕ applied to each of the coefficient of the first matrix (ie. $\boldsymbol{M_2} = \phi(\boldsymbol{M_1})$). Then we use the Chinese Remainder Theorem to find a matrix that satisfying all constraint on the remainders modulo every prime p (as a recall, the Chinese remainder theorem is used to solve congruence equations).

Gluing method **of [Cramer and Xing, 2019]**
It is shown in [Cramer and Xing, 2019] that the second monotone span program can be built with the Vandermonde matrix. Then, after obtaining the matrix that computes the access structure for any prime number $p \leq n$ and the Vandermonde matrix that compute this access structure for each prime $p \geq n$, we can glue these matrices following a specific construction in a way that the new matrix computes a desired monotone span program. As we obtained a matrix of a monotone span program computing a given access structure modulo all prime p, we can build secret sharing scheme from this matrix

2.4 Conclusion

We sum up the idea of how to build such a matrix. As the construction the BBSSS is very general, we need to do some extra work to determine how to implement the secret sharing scheme and how to instantiate it. There is often a gap between theoretical results and an application of what has been found. Since no works has be published on any attempts of such an implementation, the next step is to think about how to start it.

References

[Cramer and Xing, 2019] Ronald Cramer and Chaoping Xing. Blackbox secret sharing revisited: A coding-theoretic approach with application to expansionless near-threshold schemes. Cryptology ePrint Archive, Paper 2019/1134, 2019. https://eprint.iacr.org/2019/1134.

[Karpman, 2022] Pierre Karpman. A short introduction to secret sharing from linear codes. November 2022.

[Shamir, 1979] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979.