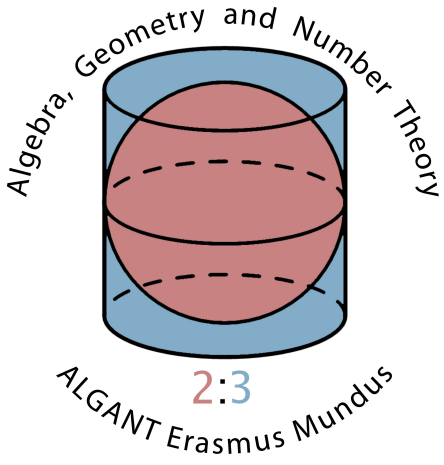
On Proving Permutations in Zero-Knowledge

Astolfi, A.

Andrea Astolfi

On Proving Permutations in Zero-Knowledge



Master thesis defended on 2011
Supervisor: Ronald Cramer

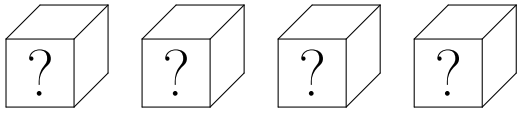
Matematisch Insituut
Leiden Universiteit

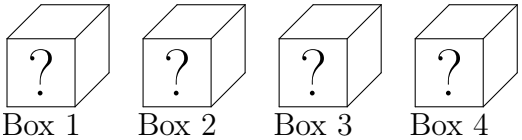
Univerisità degli Studi di Padova
Facoltà di Scienze MM.FF.NN.

Contents

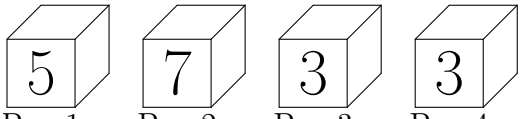
Introduction

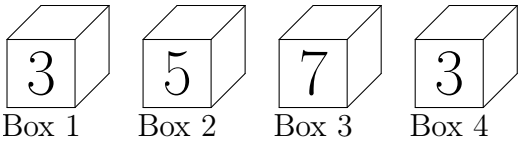
Imagine the following situation: in front of you there is a table with two rows of n locked boxes. Each box contains a piece of paper with a number written on it.



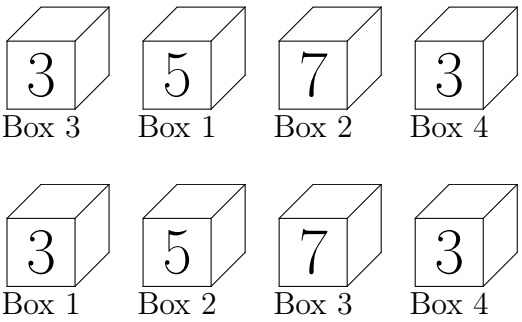


I want to convince you that the boxes of the first row hide the same numbers that are in the boxes of the second row, even if in a different order.





Moreover I want to convince you that I know how to shuffle the first row to have also the number in the same order.



This can look like a trivial problem but I add two requirements:

1. I do not want to open any boxes
2. I do not want to reveal you the shuffle

Now my task seems to be impossible. However, if we describe the same problem in mathematical terms, several solutions are possible.

The first paper that presents the idea of shuffling (or mixing), is [7] by Chaum. In this paper we also find some applications of this problem in voting and anonymous email. The question of how to verify it without revealing any other information was introduced by Sako and Kilian, in a paper about voting [8].
From those starting points, lots of protocols for verifiable mixers have been published along with many applications. For example we find the need to verify the permutation of a vector in:

- Integer comparison protocols.
- Protocols, such as Mix&Match [9] that involve mixing of truth tables of Boolean gates.
- In secure multiparty computation.
- In the contest of electronic voting.
- In anonymous communication.

In this thesis we will provide some solutions to this problem. The first uses a well know construction and it is what we want to improve. The second is a new protocol that uses permutation matrices. The third and the fourth are the most important ones. Starting from an idea of de Hoogh, Schoenmakers, Škorić and Villegas in their article about verifiable rotations [6] we generalize it for more groups of permutations, finding two protocols that are more efficient of the firsts two in terms of communication cost required.

The thesis is organized as following:

In the first chapter of this thesis we will describe the mathematical tools to create the locked boxes. This is the notion of commitment schemes. We start giving the reader some useful definitions from probability theory. Then we will define a commitment scheme and the fundamental security properties. Later we will describe how two players use a commitment scheme and how this reflects on the security properties. Finally we will give three examples based on the RSA assumption, on the Discrete Logarithm assumption and on the existence of one-way functions.

The second chapter is dedicated to describe how it is possible to prove a statement without revealing anything beyond its validity. This is the notion of zero-knowledge protocol. At the beginning we will define the very general proof system, a variant of it called proof of knowledge and the zero-knowlege property of such protocols. Then we specialize the proofs of knowledge to Σ-protocols. Finally we will describe a general construction, know as Cut and Choose, that can be used to build Σ protocols in the case that we want to prove an equivalence relation.

The third chapter is completely dedicated to describe Σ-protocols that verify relations between committed values. The basic stone will be the multiplication protocol and from this we will have a inner product protocol and a matrix multiplication protocol. Then we will also describe the OR protocol and its generalization. Finally we will prove a generalized version of the Lagrange interpolation theorem and we will use it to generalize commitment schemes and protocols over extensions of the base field on which they are defined.

The fourth chapter is the core of this thesis. Given some notion on permuta-

tions, group actions and permutation matrices, we provide a first solution to the initial problem, using the Cut and Choose construction. Then we immediately give a better solution, using permutation matrices in terms of cheating probability. Next we describe the idea provided by de Hoogh, Schoenmakers, Škorić and Villegas [6] to solve the problem restricted to rotations. Such idea consists in changing the problem from proving permutation to proving multiplication of committed values. Since protocols for this already exists (one is presented in chapter 3), the solution comes for free. The changing of domain is obtained via the Discrete Fourier Transform but we provide some notions to see it in terms of group algebra and Wedderburn decomposition. This allows us to change the group (that is, a cyclic group in case of rotations) to other groups of permutations. In our first protocol, the cardinality of such group is not modified, hence our protocol cannot be used with any permutations but just with the ones in the group of permutations set at the beginning of the protocol. At the end of the chapter, we will provide condition on such group that allows us to use bigger groups and we will describe the new protocol to do it.

In the last chapter we give some indication on the road that can be followed to improve the protocols that we described in the previous chapter through some group theory.

Chapter 1

Commitment Schemes

Introduction

In this chapter we will introduce commitment schemes, which are the digital equivalent of non-transparent locked boxes.
As motivation for the construction on commitment schemes we tell a story as it appeared in the first article talking about commitment schemes.

> Alice and Bob want to flip a coin by telephone. (They have just divorced, live in different cities, want to decide who gets the car.) Bob would not like to tell Alice HEADS and hear Alice (at the other end of the line) say "Here goes... I'm flipping the coin... You lost!" (Manuel Blum 1981, "Coin flipping by telephone, a protocol for solving impossible problems")

Making a commitment means that a player, that we will call the prover P, chooses an element in some finite set and commits to his choice towards another player, the verifier V, while keeping it secret. In a later stage the commitment is opened and it is guaranteed that the opening can yield only a single value.

A commitment scheme is a two-phase protocol between two players.

Commit phase: During this phase, P commits to the value chosen This will be the equivalent of putting a piece of paper in a non transparent box, locking the box with a combination padlock and giving the box to V.

Reveal phase: During this phase, P reveals his choice. This will be the equivalent of revealing the combination of the padlock to V.

A commitment scheme must satisfies the following properties:

Hiding property: At the end of the first phase, the verifier does not gain any knowledge of the prover's value. This has to hold even if the verifier tries to cheat.

Binding property: There exists at most one value that the verifier can accept in the second phase as legal "opening" of the commitment. This has to hold even if the prover tries to cheat.

In a commitment scheme it is sometimes necessary to run also a set-up phase. This is the equivalent of choosing the proper box. This phase may not be repeated every time that the protocol is run, but can be done once and for all. The reason for that and the description of this phase will be clear later in this chapter.

1.1 Some probability definitions

In order to give definitions of hiding and binding properties and in general of commitment schemes, we have to formalized some concepts that come from probability theory. For a close view of this topic we suggest the reading of the sixth chapter of Shoup's book. [11]

We recall that a probabilistic algorithm $\mathcal{P}$ is an algorithm that takes the regular input x and an extra value r selected at random from a finite set and independent from x. The output y depends on both x and r. In particular given x, the output $y = \mathcal{P}(x)$ is a random variable and

$$\text{Prob}[\mathcal{P}(x) = y] \;=\; \text{Prob}[r \in R_{x,y}]$$

where $R_{x,y} = \{r | \mathcal{P}_D(x, r) = y\}$ and $\mathcal{P}_D(x, r)$ is the deterministic algorithm that has the same output of $\mathcal{P}$ on input x and random choice r.

Definition 1.1.1 (Probability distributions)**.** *A finite probability distribution $U = (\mathcal{U}, \mathbf{P})$ is a finite non-empty set $\mathcal{U}$ together with a function $\mathbf{P}$ that maps $u \in \mathcal{U}$ to $\mathbf{P}(u) \in [0, 1] \subset \mathbb{R}$ such that*

$$\sum_{u \in \mathcal{U}} \mathbf{P}(u) = 1.$$

The set $\mathcal{U}$ is called the sample space and the function $\mathbf{P}$ is called the probability function.

In this thesis we shall use the phrase "probability distribution" to mean "finite probability distribution".

Definition 1.1.2 (Family of probability distribution)**.** *A family of probability distributions $V = (\mathcal{X}, \{V_x\}_{x\in\mathcal{X}})$ consists of a set $\mathcal{X}$ called index set and a corresponding set $\{V_x\}_{x\in\mathcal{X}}$ of probability distributions defined over the same sample space.*

Definition 1.1.3 (Negligible Function)**.** *A function $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ is called negligible in ℓ if $\forall c \in \mathbb{N}$ there exists an integer ℓ_c such that $\forall \ell \geq \ell_c$*

$$\varepsilon(\ell) \leq \ell^{-c}$$

The previous definition is motived by the following: if an event occurs with negligible probability $\varepsilon(l)$ then we expect that such event happens after $\frac{1}{\varepsilon(l)}$ repetitions. Hence this event happens so seldom that we may assume that probabilistic polynomial time algorithms will never produce it.

Definition 1.1.4 (Infeasible)**.** *A computational task is infeasible in ℓ if, for every probabilistic polynomial time algorithm, the probability of completing the computation is negligible in ℓ.*

The next concept that we will introduce will be the one of statistical distance. This will allow us to compare two different probability distributions.

Definition 1.1.5. *Given two probability distributions $\mathcal{P} = (\mathcal{Y}, \mathbf{P})$ and $\mathcal{Q} = (\mathcal{Y}, \mathbf{Q})$, we define the statistical distance between them as*

$$\mathrm{SD}(\mathcal{P}, \mathcal{Q}) = \sum_{y\in\mathcal{Y}} |\mathbf{P}(y) - \mathbf{Q}(y)|$$

Definition 1.1.6 (Perfectly Indistinguishable)**.** *Given two families of probability distribution $U = (\mathcal{X}, \{U_x\}_{x\in\mathcal{X}})$ and $V = (\mathcal{X}, \{V_x\}_{x\in\mathcal{X}})$, we define them as perfectly indistinguishable if $U_x = V_x$ for all $x \in \mathcal{X}$. We will write it as $U \sim^p V$*

Definition 1.1.7 (Statistically Indistinguishable)**.** *Given two families of probability distribution $U = (\mathcal{X}, \{U_x\}_{x\in\mathcal{X}})$ and $V = (\mathcal{X}, \{V_x\}_{x\in\mathcal{X}})$, we define them as statistically indistinguishable if for any $x \in \mathcal{X}, \mathrm{SD}(U_x, V_x)$ is negligible in the binary length of x. We will write it as $U \sim^s V$*

We require that the statistical indistinguishability holds for all the possible indexes, however it is also possible to give a weaker definition in which the requirement has to hold for all but a negligible fraction of the indexes.

In order to define a computational flavor of indistinguishability, we need to introduce a party that tries to distinguish between the probability distributions. Suppose that we do the following experiment.

Given two probabilistic family $U = (\mathcal{X}, \{U_x\}_{x\in\mathcal{X}})$ and $V = (\mathcal{X}, \{V_x\}_{x\in\mathcal{X}})$ and a probabilistic polynomial time algorithm $\mathcal{G}$ that on input 1^ℓ output an index $x \in \mathcal{X}$ of binary length ℓ, we choose one of the two probability distributions corresponding to x (U_x or V_x) and we call it y. Then we give x and y to a third party, called distinguisher, and we ask him to guess if $y = U_x$ or $y = V_x$. We call the advantage of a distinguisher the absolute value of the difference between the probability of a distinguisher of correctly guessing the family (over the random choice of $\mathcal{G}$) and $\frac{1}{2}$.

Definition 1.1.8 (Computational Indistinguishable)**.** *Given two families of probability distribution $U = (\mathcal{X}, \{U_x\}_{x\in\mathcal{X}})$ and $V = (\mathcal{X}, \{V_x\}_{x\in\mathcal{X}})$, we define them as computational indistinguishable if, given any probabilistic polynomial time distinguisher D, the advantage of D over the random choice of $\mathcal{G}$ is a negligible function in ℓ. We will write it as $U \sim^c V$*

1.2 Definition of commitment schemes

Definition 1.2.1 (Family of functions)**.** *A family of functions $F = (I, \{f_i\}_{i\in I})$ is a set $I \subset \{0,1\}^*$, called index set and a corresponding set $\{f_i\}_{i\in I}$ of finite functions. That is, for each $i \in I$, the domain of the function f_i, denoted D_i, is a finite set.*

We are now ready to define a commitment scheme.

Definition 1.2.2 (Commitment scheme)**.** *A commitment scheme is a probabilistic polynomial time algorithm, denoted by $\mathcal{G}$ and called a key generator,*

together with two families of functions $\texttt{commit} = (\mathcal{PK}, \{\texttt{commit}_{pk}\}_{pk\in\mathcal{PK}})$ *and* $\texttt{verify}(\mathcal{PK}, \{\texttt{verify}_{pk}\}_{pk\in\mathcal{PK}})$ *that are efficiently computable. They have to satisfy the following:*

Key generator: *The key generator* $\mathcal{G}$ *takes as input a security parameter* 1^ℓ *an output and element* $pk \in \mathcal{PK}$*. We will refer to this element as to the public key.*

Functions: *Given two values* ℓ_r *and* ℓ_b *polynomially bounded in* ℓ *(they can also depend on* pk*) we have:*

$$\texttt{commit}_{pk} : \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_b} \to \{0,1\}^{\ell}$$

$$\texttt{verify}_{pk} : \{0,1\}^{\ell_r} \times \{0,1\}^{\ell_b} \times \{0,1\}^{\ell} \to \{0,1\}$$

Even if the definition allows to choose families of functions that are not biding or not hiding, such commitment would not be interesting. Hence we always require that a commitment scheme satisfies:

Hiding: Given $pk \leftarrow \mathcal{G}(1^\ell)$, $\forall b_0, b_1 \in \{0,1\}^{\ell_b}$, the probability distributions $\texttt{commit}_{pk}(r, b_0)$ and $\texttt{commit}_{pk}(s, b_1)$ are indistinguishable for random independent r, s in $\{0,1\}^{\ell_r}$.

Binding Given $pk \leftarrow \mathcal{G}(1^\ell)$, compute C, r, b, r', b' with $b \neq b'$ such that

$$\texttt{verify}_{pk}(r, b, C) = \texttt{verify}_{pk}(r', b', C) = 1$$

is infeasible in ℓ.

In section 1.1 we described different flavors of indistinguishability. This reflects in different flavors of the hiding and binding properties. In particular we have:

Definition 1.2.3 (Unconditionally Binding)**.** *A commitment scheme is unconditionally binding if given* $pk = \mathcal{G}(1^\ell)$*, for every* C*, if there exists two pair* (r, b) *and* (r', b') *such that*

$$\texttt{verify}_{pk}(r, b, C) = 1 = \texttt{verify}_{pk}(r', b', C) = 1 = .$$

Then we have $b = b'$*.*

Definition 1.2.4 (Unconditionally Hiding)**.** *A commitment scheme is unconditionally hiding if for all possible* $pk \leftarrow \mathcal{G}(1^\ell)$, $\forall\, b_0, b_1 \in \{0,1\}^{\ell_b}$

$$\texttt{commit}_{pk}(r, b_0) \sim^s \texttt{commit}_{pk}(s, b_1)$$

with random independent r, s*.*

Definition 1.2.5 (Computationally Binding)**.** *A commitment scheme is computationally binding if, taking any probabilistic polynomial time algorithm* P^* *on input* pk*, the probability* $\varepsilon(l)$ *(over the random choice of* P^* *and* $\mathcal{G}$*) that* P^* *outputs* C, b, r, b', r' *such that* $b \neq b'$ *and*

$$\texttt{verify}_{pk}(r, b, C) = 1 = \texttt{verify}_{pk}(r', b', C)$$

is negligible in ℓ*.*

Definition 1.2.6 (Computationally Hiding)**.** *A commitment scheme is computationally hiding if* $\forall b_0, b_1 \in \{0,1\}^{\ell_b}$

$$(pk, \texttt{commit}_{pk}(r, b_0)) \sim^c (pk, \texttt{commit}_{pk}(s, b_1))$$

with random independent r, s *and* $pk \leftarrow \mathcal{G}(1^\ell)$*.*

Now we would like to have unconditionally binding and unconditionally hiding commitment schemes. However this is impossible. In fact suppose that we have such a scheme. Now consider the probability distributions

$$U_{b_0} = \texttt{commit}(r, b_0) \text{ and } U_{b_1} = \texttt{commit}(s, b_1)$$

for random independent r, s and $b_0 \neq b_1$. By the binding property we have that for all possible event C,

$$|\operatorname{Prob}[U_{b_0} = C] - \operatorname{Prob}[U_{b_1} = C]| = 1,$$

that means that one of the two probabilities is 1 and the other 0. Indeed suppose that exists a C such that the two probabilities are positive, this means that

$$\texttt{commit}(r, b_0) = \texttt{commit}(s, b_1) = C$$

and hence

$$\texttt{verify}(r, b_0, C) = \texttt{verify}(s, b_1, C) = 1$$

which is a contradiction.
Now from $|U_{b_0}(C) - U_{b_1}(C)| = 1$ it is clear that the statistical distance is not negligible in ℓ and we have again a contradiction.
So we have just two main types of commitment schemes: unconditionally hiding and computationally binding or unconditionally binding and computationally hiding.

Observation 1.2.7. In the definition 1.2.2 we do not speak at all about players and interactions. To understand the reason think at the following: the definition of commitment scheme is the same as the description of a non-trasparent locked box. Now all the properties come from the quality of the box, not from how you use it. The same happens with commitment scheme. The properties of hiding and binding are satisfied by the families of functions `commit` and `verify`. However we will here describe how a prover P and a verifier V can use the key generator and the families of functions and from now on we will intend that a commitment schemes is always used in the following way:

Set-up phase: The set-up phase is performed by running $\mathcal{G}(1^\ell)$. This can be done by one of the two players (and this reflects on the flavor of hiding and binding properties) or by a trusted third party. In any case both the player must be convinced that the public key is correctly generated, namely that $pk \in \mathcal{PK}$.

Commit phase: To commit a value $b \in \{0,1\}^{\ell_b}$ P chooses at random $r \in \{0,1\}^{\ell_r}$ and computes $C = \texttt{commit}_{pk}(r, b)$

Verify phase: To verify a legal opening V checks that $\texttt{verify}_{pk}(r, b, C) \stackrel{?}{=} 1$

In general in the commitment schemes presented in this thesis, it can always be checked explicitly that $pk \in \mathcal{PK}$ but in other cases some zero-knowledge protocols (see chapter 2) must be used. Moreover we observe that, to have an unconditionally binding scheme, the generator must be run by P while to have an unconditionally hiding scheme, it must be run by V. Finally when the public key is clear from the context we will use the notation $\mathrm{Com}(r, b)$ instead of $\texttt{commit}_{pk}(r, b)$.

A final remark on the communication is the following: when dealing with those interactions, we always assume that no mistake can happen during the

communication between the players. If the assumption of "perfect" communication (communication without errors) is substituted with different assumptions, such as the presence of a noisy channel or of a network in which a bounded number of parties have been corrupted, a lot of different scenarios are possible. The commitment schemes must be defined in a different way and very different results hold, for example there are scenarios in which is possible to have commitment schemes both unconditionally hiding and unconditionally binding even if with an higher cost in terms of communication cost. However this goes very far from the purpose of this thesis. For a deeper treatment of the topic see [15], [16] and [17]:

1.3 Examples

Now we will give some example of commitment schemes based on functions that are "hard" to invert. We refer to [10] for an exhaustive and formal treatment of the one-way functions.

Definition 1.3.1 (One-way function)**.** *We say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function if the following holds:*

Easy to compute: *There exists a polynomial time algorithm A which on input x, outputs $f(x)$*

Hard to invert: *For every probabilistic polynomial time algorithm A^* on inputs 1^ℓ and $f(U_\ell)$ where U_ℓ is a random variable with uniform distribution over $\{0,1\}^\ell$, the probability that the output of A^* is in $f^{-1}(f(U_\ell))$ is negligible in ℓ.*

Definition 1.3.2 (Family of one-way functions)**.** *A family of functions $F = (I, f_i : D_i \to \{0,1\}^*)$ is called one way if there exist three probabilistic polynomial time algorithm $\bar{I}, \bar{D}, F$ such that:*

Easy to compute: *$\bar{I}$ takes as input 1^ℓ and outputs an index of the set $I \cap \{0,1\}^\ell$; we will denote with I_ℓ the random variable describing his output.*
$\bar{D}$ takes as input $i \in I \cap \{0,1\}^\ell$ and output a random element $x \in D_i$; we will denote with X_ℓ the random variable describing his output.
F takes as inputs $i \in I \cap \{0,1\}^\ell$ and $x \in D_i$ and always outputs $f_i(x)$.

Hard to invert: *For every probabilistic polynomial time algorithm A^* on inputs 1^ℓ and $f_{I^\ell}(X_\ell)$, the probability over the outputs of $\bar{I}$ and $\bar{D}$ that the output of A^* is in $f_{I_\ell}^{-1}(f_{I_\ell}(X_\ell))$ is negligible in ℓ.*

We recall that no family of functions has been proved to be one-way, however some of them are believed to be so. In the cryptography context some family functions are assumed to be one-way and the security of several cryptographic primitives, including commitment schemes, depends on this assumptions.

1.3.1 RSA-based example

The commitment scheme that we are going to describe was firstly presented as an identification scheme based on the RSA cryptosystem ([19]).

Definition 1.3.3. *An RSA-generator is a probabilistic polynomial time algorithm $\mathcal{H}$ that, on input 1^ℓ, outputs $n, e \in \mathbb{N}$ such that:*

- *$n = p \cdot q$ with p and q prime independently chosen at random such that the binary length of n is ℓ.*
- *e chosen at random such that $1 < e < \varphi(n)$ and $\gcd = (e, \varphi(n)) = 1$*

Assumption 1.3.4 (RSA assumption)**.** *Given the RSA-generator $\mathcal{H}$ as in definition 1.3.3 on input 1^ℓ, the family of functions $f = (I, \{f_i\}_{i \in I})$ where $I = \{(n, e) | (n, e) \leftarrow \mathcal{H}\}$ and, for $i = (n, e)$*

$$\begin{aligned} f_i : \ \mathbb{Z}_n^* &\to \mathbb{Z}_n^* \\ x &\to x^e \mod n \end{aligned}$$

is a one-way family of functions.

Lemma 1.3.5. *Given an output (n, e) of the RSA generator $\mathcal{H}$ as in definition 1.3.3 on input 1^ℓ and $r, y, c \in \mathbb{Z}_n^*$ with $0 < c < e$ such that:*

$$r^e = y^c \pmod n$$

it is possible to compute the unique $x \in \mathbb{Z}_n^*$ such that $x^e = y \pmod n$ in polynomial time in ℓ.

Proof. In this proof (and in general in this section) we will use the symbol $\equiv$ to mean the equality in $\mathbb{Z}_n^*$ i.e. the equality mod n.
If c and e are coprime, we can find s and t such that $sc = 1 + te$ with the Bezout identity (that is computable in polynomial time using the Euclidian algorithm), and

$$y^c \equiv (r)^e \Rightarrow y^{sc} \equiv (r^s)^e \Rightarrow y^{1+te} \equiv (r^s)^e \Rightarrow y \equiv (r^s y^{-t})^e$$

Which implies $x \equiv r^s y^{-t}$ since e is coprime with $\varphi(n)$ and hence the e-roots are unique.
If $\gcd(c, e) = m$ with $m > 1$, we can divide e and c by m obtaining e' and c' and a new equation:

$$y^{c'} \equiv r^{e'}$$

since m is coprime with $\varphi(n)$.
Now we can apply the same procedure as before to compute x (Observe that e' is still coprime with $\varphi(n)$). □

Commitment Scheme 1.3.6 (RSA based commitment scheme)**.**

Key generator: *The key generator $\mathcal{G}$ is the RSA-generator $\mathcal{H}$ as in definition 1.3.3 that takes as input 1^ℓ, extended by making it choosing $x \in \mathbb{Z}_n^*$ at random and computing $y = x^e \mod n$. The output is $pk = (n, e, y)$*

Commit functions: *The functions of the family* `commit` *are defined as:*

$$\begin{aligned} \texttt{commit}_{pk} : \ \mathbb{Z}_n^* \times \{0,1\}^{\ell_b} &\to \mathbb{Z}_n^* \\ (r, b) &\to y^b r^e \mod n \end{aligned}$$

Verify functions: *The verify functions of the family* `verify` *are defined as:*

$$\begin{aligned} \texttt{verify}_{pk} : \ \mathbb{Z}_n^* \times \{0,1\}^{\ell_b} \times \mathbb{Z}_n^* &\to \{0,1\} \\ (r, b, C) &\to \begin{cases} 1 \text{ if } y^b r^e \equiv C \\ 0 \text{ if } y^b r^e \not\equiv C \end{cases} \end{aligned}$$

Proposition 1.3.7. *The commitment scheme 1.3.6 is unconditionally hiding and computationally binding assuming 1.3.4.*

Proof.

Unconditionally Hiding: For every $pk \leftarrow \mathcal{G}(1^\ell)$, the distribution of the commitment is independent of b. This is due to the fact that r is chosen uniformly at random so r^e has also the uniform distribution. Thus for all b, $y^b r^e$ has uniform distribution over $\mathbb{Z}_n^*$ since y^b is a constant (we are considering $y^b r^e$ as a function of r).

Computationally Binding: The biding property is computational satisfied. Indeed suppose by contradiction that we have a probabilistic polynomial time algorithm A that on input $n, q, y, c = \mathrm{Com}(r, b)$ outputs with a non negligible probability r_0, r_1, b, b' such that $b \neq b'$ and

$$y^b r_0^e \equiv \mathrm{Com}(r_0, b) \equiv c \equiv \mathrm{Com}(r_1, b') \equiv y^{b'} r_1^e$$

Then we have that

$$(r_0 r_1^{-1})^e \equiv r_0^e r_1^{-e} \equiv y^{-b} c y^{b'} c^{-1} \equiv y^{b'-b}$$

hence by lemma 1.3.5 (recalling that b and b' are smaller than e) we can find x in polynomial time with non negligible probability which is against the RSA assumption.

□

Finally if we use this commitment scheme as described in observation 1.2.7, then V has to run $\mathcal{G}$ and he sends the n, e, y to P. P has to check that

$$y \equiv a^e \text{ for some } a \in \mathbb{Z}_n^*.$$

This can be done in polynomial time checking that $\gcd(y, n) = 1$ with the Euclidian algorithm. The fact that n and e are correctly generated cannot be checked so easily, however this is not necessary. In fact the hardness of inverting the commit function is a guarantee of the binding property, hence V has all the interest in generate n and e "as hard as possible" and hence correctly (according to definition 1.3.3).

1.3.2 Discrete logarithm based example

The commitment scheme that we are going to describe is based on the Discrete Logarithm problem. It takes the name of Pedersen commitment by the first author that describe it ([18]).

Definition 1.3.8. *We define as DLP-generator a probabilistic polynomial time algorithm $\mathcal{H}$ that, on input 1^ℓ, outputs $q \in \mathbb{N}, g,$ `lab` such that:*

- *q is a prime such that $2^{\ell-1} < q < 2^\ell$.*
- `lab` *is a label to "a group G of order q" (it will be clarified later).*

- *g is random generator of G.*

Moreover we require that the binary representation of an element of G is always unique and that the binary representation of any elements of the group G is in $\{0,1\}^{M_G(\ell)}$ for $M_G(\ell)$ depending on G and polynomially bounded in ℓ.

The reason for requesting $\mathcal{G}$ to output a label instead of the group is that we want to keep the group implicit, i.e. we do not want to give a mathematical description of such a group. However we want to be able to run algorithms that take as input such a group and we will do it giving them the label that points to the group.
To be able to efficiently compute the function `commit` and `verify` we have to assume the existence of two algorithms that take as common inputs a prime q and a label `lab` of a group G of order q.

- The algorithm `mult` that on inputs, $q, \texttt{lab}, h, g$ computes in polynomial time $g \cdot h$ in the group G pointed by the label `lab`
- The algorithm `isin` that on inputs, $q, \texttt{lab}, h$ outputs in polynomial time, 1 if $h \in G$ and 0 if $h \notin G$

Those algorithms allows us not only to compute the function `commit` and `verify` but to do all the operation in G, knowing just `lab`. In fact from the multiplication algorithm it is possible to compute efficiently g^x for any$x \in \mathbb{Z}_p$ by computing $g^2, g^4 = g^2 \cdot g^2, g^8 = g^4 \cdot g^4 ...$ until g^{2^ℓ}, considering the binary representation of $x = (x_0, \cdots, x_\ell)$ and computing

$$g^x = \prod_{i=0}^{2^\ell} g^i x_{\ell - i}$$

Assumption 1.3.9 (DLP assumption)**.** *Given the DLP-generator $\mathcal{H}$ as in definition 1.3.8 on input 1^ℓ the family of functions $f = (I, \{f_i\}_{i \in I})$ where $I = \{(q, g, \texttt{lab}) | (q, g, \texttt{lab}) \leftarrow \mathcal{H}\}$ and, for $i = (q, g, \texttt{lab})$*

$$\begin{aligned} f_i : \ \mathbb{Z}_q &\to G \\ x &\to g^x \end{aligned}$$

is a one-way family of functions.

Commitment Scheme 1.3.10 (Pedersen commitment)**.**

Key generator: *The key generator $\mathcal{G}$ is the generator $\mathcal{H}$ as in definition 1.3.8 that takes as input 1^ℓ extended as follow: it also chooses x at random in G and computes $h = g^x$. The output is $pk = (q, g, h, \texttt{lab})$.*

Commit functions: *The functions of the family* `commit` *are defined as:*

$$\begin{aligned} \texttt{commit}_{pk} : \ \mathbb{Z}_q \times \mathbb{Z}_q &\to G \\ (r, b) &\to h^b g^r \end{aligned}$$

Verify functions: *The verify functions of the family* `verify` *are defined as:*

$$\begin{aligned} \texttt{verify}_{pk} : \ \mathbb{Z}_q \times \mathbb{Z}_q \times G &\to \{0,1\} \\ (r, b, C) &\to \begin{cases} 1 \text{ if } h^b g^r = C \\ 0 \text{ if } h^b g^r \neq C \end{cases} \end{aligned}$$

Proposition 1.3.11. *The commitment scheme 1.3.10 is unconditionally hiding and computationally binding assuming 1.3.9.*

Proof.

Unconditionally Hiding: For every $pk \leftarrow \mathcal{G}(1^\ell)$, the distribution of the commitment is independent of b. This is due to the fact that r is chosen uniformly at random so g^r has also the uniform distribution over G, Thus for all b, $h^b g^r$ has uniform distribution over G since h^b is a constant (we are considering $h^b g^r$ as a function of r).

Computationally Binding: Suppose by contradiction that we have a probabilistic polynomial time algorithm A that on input $\texttt{lab}, q, g, h, c$, where $c = \mathrm{Com}(r, b)$, outputs with a non negligible probability r_0, r_1, b, b' such that $b \neq b'$ and (in the group G)

$$h^b g^{r_0} = \mathrm{Com}(r_0, b) = c = \mathrm{Com}(r_1, b') = h^{b'} g^{r_1}$$

Then we have that

$$h^b g^{r_0} = h^{b'} g^{r_1} \Rightarrow g^{r_0 - r_1} = h^{b' - b}$$

that implies

$$r_0 - r_1 = x(b - b') \pmod q$$

hence, since q is prime, we can invert $(b-b')$ and compute $x = \log_g h$ in polynomial time (because we are using the algorithm `mult`) with non negligible probability which is against the DLP assumption.

□

Finally if we use this commitment scheme as described in observation 1.2.7, then V has to run $\mathcal{G}$ and he sends $q, g, h, \texttt{lab}$ to P. P has to check that q is a prime and that g and h are in G. This can be done with a polynomial time primality test and with the algorithm `isin`. The fact that g is a generator of G and that G is a group for which extract the discrete logarithm is hard cannot be checked easily, however this is not necessary. In fact V has all the interest in choosing G "as hard as possible" and hence correctly (thus according to definition 1.3.8).

1.3.3 General one-way function based approach

In general if we assume the existence of a family of 1-1 one-way functions, one can define from it a commitment scheme with unconditionally binding and computationally hiding. We report here such construction but we give reference to [10] for the proofs of the properties:

Definition 1.3.12 (One-Way function based commitment scheme)**.** *Given a family of 1-1 one-way functions $F = (I, f_i)$ that is f_i is injective for all $i \in I$, with domain $D_i = \{0,1\}^{n_i}$ we can define a commitment scheme as follows:*

Key generator: *The key generator $\mathcal{G}$ on input 1^ℓ outputs $pk = (i)$ such that $i \in I$ and $D_i = \mathbb{F}_2^{\ell_i}$ where ℓ_i is polynomially bounded in ℓ*

Commit function: *The functions of the family* `commit` *are defined as:*

$$\begin{aligned} \texttt{commit}_{pk}: \ \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2 &\to \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2 \\ ((r,x),b) &\to ((r, f(x)), (x \cdot r) \oplus b) \end{aligned}$$

where $\cdot$ is the inner product mod 2 *and $\oplus$ the component-wise sum* mod 2

Verify function: *The verify functions of the family* $\texttt{verify} =$ *are defined as:*

$$\texttt{verify}_{pk}: \quad \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2 \times \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2 \to \mathbb{F}_2$$

$$((r,x),b,C,D,E) \to \begin{cases} 1 \ \textit{if} \ r = C \\ \qquad f(x) = D \\ \qquad (f(x) \cdot r) \oplus b = E \\ 0 \ \textit{else} \end{cases}$$

Such commitment scheme is a computationally hiding and unconditionally binding. A proof can be found in [10] (Construction 4.4.2 and Proposition 4.4.3) observing that $x \cdot r$ is an hard-core predicate of $g(x,r) = (f(x),r)$ (again [10], Definition 2.5.1 and Theorem 2.5.2).
Moreover it can be easily generalized to a commitment scheme bit-strings in the following way:
Suppose that $b \in \{0,1\}^{n_\ell}$, then we can obtain a random committed value $s \in \{0,1\}^{n_\ell}$ computing $(x \cdot r)$, for the first bit of s, $(f(x) \cdot r)$ for the second and so on until $f^{n_\ell - 1}(x) \cdot r$ for the last. Then the commit function would be:

$$\begin{aligned} \texttt{commit}_{pk}: \ \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{n_\ell} &\to \mathbb{F}_2^{\ell_i} \times \mathbb{F}_2^{n_\ell \ell_i} \times \mathbb{F}_2^{n_\ell} \\ ((r,x),b) &\to (r, f(x), \ldots, f^{n_\ell}(x), s \oplus b) \end{aligned}$$

Another general approach, more similar to the Pedersen commitment can be found in [4], and it is based on one-way group homomorphism.

1.4 Homomorphic commitment schemes

In order to use the commitment schemes as parts of more general protocols is often useful to require some algebraic structures on the domain and codomain of the `commit` functions. We do not define here what an homomorphic commitment is, but we proof some useful properties of the Pedersen commitment (example 1.3.10). In general in the thesis we will assume to be using Pedersen commitment even if other commitment schemes with similar properties do exist.

Proposition 1.4.1. *Suppose we have the Pedersen commitment scheme (example 1.3.10). Suppose also that, given a public key pk,*

$$\texttt{verify}_{pk}(r_a, a, A) = 1 \ \textit{and} \ \texttt{verify}_{pk}(r_b, b, B) = 1$$

with $r_a, a, r_b, b \in \mathbb{Z}_q$ *and* $A, B \in G$. *Then*

$$\texttt{verify}_{pk}(r_a + r_b, a + b, A \cdot B) = 1$$

$$\texttt{verify}_{pk}(\lambda r_a, \lambda a, A^\lambda) = 1$$

Proof. We have in G:

$$A{\cdot}B = \mathrm{Com}(r_a, a){\cdot}\mathrm{Com}(r_b, b) = g^{r_a}h^a{\cdot}g^{r_b}h^b = g^{r_a+r_b}h^{a+b} = \mathrm{Com}(r_a{+}r_b, a{+}b)$$

hence $\texttt{verify}_{pk}(r_a + r_b, a + b, A \cdot B) = 1$
Moreover

$$A^\lambda = (g^{r_a}h^a)^\lambda = g^{\lambda r_a}h^{\lambda a} = \mathrm{Com}(\lambda r_a, \lambda a)$$

hence $\texttt{verify}_{pk}(\lambda r_a, \lambda a, A^\lambda) = 1$

□

Chapter 2

Zero Knowledge Protocols

Introduction

"Is it possible to prove a statement without revealing anything beyond this fact?".

Consider the following motivational example:

Example 1. There are two rooms. Every room has its own entrance and there is a door connecting them. This door can be opened inserting the right secret code. Peggy claims that she knows this code and wants to sell it to Victor. Clearly Peggy doesn't want to reveal the code before receiving the money from Victor but Victor doesn't want to pay Peggy before being sure that the code is correct. Hence they need a way (a protocol) to verify that the code is correct, without forcing Peggy to reveal it.

This example give rise to a first and very informal definition of zero-knowledge: *a protocol is zero-knowledge if it communicates exactly the knowledge that was intended and no (zero) extra knowledge.*

For a discussion of what a "proof" is and all the possible interpretations of the phrase "without revealing anything beyond this fact" we refer to [10]. In this chapter we will introduce a formal definition of proof systems, the notion of zero-knowledge as an additional property of them and finally we will describe a particular class, the Σ-protocols, that we will use in this thesis.

2.1 Proof systems

A proof system is a protocol between two entities, the prover and the verifier, which aim is to verify the truth of a statement. To check the validity of this verification procedure we request a proof system to fulfill two properties:

Completeness: If the statement is true, the verifier will accept it with "high" probability.

Soundness: The probability that a verifier will accept a false statement is "low".

2.1.1 Preliminaries

Turing machines

In order to give a formal definition of proof system, and in general of a protocol, we will model the prover and the verifier with two probabilistic, interactive Turing machines (PITMs). We will now give an informal description of it. For a formal definition see [10].
Turing machines can be seen as an abstraction of computers. To perform a computational task, the machine is provided with an sufficient number of instructions (algorithm) that are chosen from a set of eligible instructions.
The machine has also a knowledge tape that is a part of the memory and that contains any possible input.
To write and read intermediate results the machine is furnished with an auxiliary tape.
When the computation (as dictated by the algorithm) has been complete, the machine write the outcome on the output tape.
A *probabilistic* machine is also provided with a random tape which is a part of the memory allocated for the storage of random bits. The machine is allowed to read from this random tape and to use the random bits in any of its computations.
To enable the *interaction* between two probabilistic Turing machines we provide a communication tape that connects them. They can communicate by writing on and reading from this tape. If they have a common input then this is written on both the knowledge tapes.
A Turing machine is *efficient* (or polynomial time) if the total number of read and write operations is polynomially bounded in the length of the input.

Notation

Let P and V be a pair of PITMs linked together with common input x. We will indicate with $(P, V)(x)$ the random variable representing the output of V on common input x. We will recall that the output of V is interpreted as the decision on whether to accept or reject the protocol. It outputs 1 if the protocol is accepted and 0 if is rejected

Language

Every proof system is defined for a language L. A language L is a subset of the set $\{0,1\}^*$. L is the set of true statement and the input x is the statement that we want to prove. Thus if $x \in L$ then we want $(P, V)(x) = 1$ with "high" probability (completeness), while if $x \notin L$ we want $(P, V)(x) = 1$ with "low" probability (soundness).

2.1.2 Definition

Definition 2.1.1 (Interactive Proof System)**.** *The pair of probabilistic interactive Turing machines (P, V) is an interactive proof system for a language L if machine V is polynomial time and the following two condition are satisfied:*

Completeness: *For every $x \in L$ then*

$$\text{Prob}[(P, V)(x) = 0]$$

is negligible in the binary length of x

Soundness: *For every $x \notin L$ and every interactive machine P^**

$$\text{Prob}[(P^*, V)(x) = 1]$$

is negligible in the binary length of x

We want to remark that, while we require that the verifier is an efficient machine, no request has been made on the computational power of P. This means that the soundness condition is rather strong since it has to holds for all the possible interactive machine (regardless to their efficiency).

2.2 Zero-Knowledge

Loosely speaking we say that an interactive proof system (P, V) for a language L is zero-knowledge if whatever can be efficiently computed after interacting with P on input $x \in L$ could be efficiently computed from x without any interaction. This is verified by requesting that the conversation between P and any possible verifier V^* can be simulated without interacting with P. The fact that this request is made for every verifier means that we can think to the zero-knowledge as a property of P only. Thus a property that every PITM A can have and it reflects the hardness of extracting information while interacting with A.

Again, as in the commitment schemes, the different flavors of indistinguishability reflects on different flavors of zero knowledge. In particular we have:

Definition 2.2.1 (Zero-Knowledge)**.** *Let (P, V) be an interactive proof systems for some language L. We say that (P, V) is computational (statistical, perfect) zero-knowledge if, for every probabilistic polynomial time interactive machine V^*, there exists a probabilistic expected polynomial time algorithm M^* (called simulator) such that the two families of probability distributions:*

- $(P, V^*) = (L, (P, V^*)(x))$
- $M^* = (L, M^*(x))$

are computational (statistical, perfect) indistinguishable.

The implementation of the simulator may be very difficult, since it has to hold for every verifier. However there exists a much weaker definition of zero-knowledge that asks for the existence of a simulator for V only. This is called the honest verifier zero-knowledge property (HVZK). Clearly the "honest verifier" part of the name comes from the fact that we require the simulation just for V, hence just for a verifier that follows the protocol (and thus honest in an human context). Even in this case still make sense requiring that no information is revealed. In fact we can have an "honest but curious" verifier, that is a machine that follows the protocol (just as the machine V of the definition) but that tries to extract all the possible information from the interaction with P.

Definition 2.2.2 (Honest Verifier Zero-Knowledge)**.** *Let (P, V) be an interactive proof systems for some language L. We say that (P, V) is honest*

verifier zero-knowledge if there exists a probabilistic expected polynomial time algorithm M such that the two families of probability distributions:

- $(P, V) = (L, (P, V)(x))$
- $M = (L, M(x))$

are computationally indistinguishable.

2.3 Σ-protocols

This type of protocol were firstly presented in [5] by Ronald Cramer.

2.3.1 Preliminares

Relation and Witness

We will define a Σ-protocol for relation R.

Definition 2.3.1. *We said that R is a binary relation if R is a subset of $\{0,1\}^* \times \{0,1\}^*$ where the only restriction is that if $(x, w) \in R$ then the binary length of w is at most polynomial in the binary length of x.*

Notation 2.3.2. For any string x we denote by:

- $RW(x)$ the set $\{w \in \{0,1\}^* | (x, w) \in R\}$,
- RX the set $\{x \in \{0,1\}^* | RW(x) \neq \emptyset\}$

Given a relation R and $(x, w) \in R$ in a Σ-protocol the element x is the common input to P and V while the element w, that we will call witness, is a private input of P only.

Proof Systems and Proofs of Knowledge

There is a variant of proof systems, known as proofs of knowledge. The Σ-protocols that we are going to describe belong to such variant. However we will give an informal description of this and we refer to [2] for a formal definition.

While a proof system is defined for a language L and the (computationally unbounded) prover claims that the common input x is in the language, a proof of knowledge is defined for a relation R and the (computationally bounded) prover claims to know a certain piece of information. In particular if x is the common input, the prover claims to know w such that $(x, w) \in R$.

Prover and Verifier

Both the prover P and the verifier V are modeled by probabilistic polynomial time interactive Turing machines. This is a great difference with the generic definition of proof systems. The prover is here forced to be polynomial time. This explain why we have to provide P with the witness w. In fact without any advantage on V the interaction with P would be meaningless, since V would be able to accomplish any tasks of P.

Knowledge extractor

As it happens with the proof system, the notion of knowledge is strictly related, in this context, to the computational power. More precisely, we say that a protocol is a proof of knowledge if there exists a probabilistic expected polynomial time K, called knowledge extractor, that, on input x and "rewindable black-box" access to the prover, can efficiently compute w such that $(x, w) \in R$.
When we say that K has rewindable black-box access to the prover we mean the following:

Black-box: K cannot see the memory or the random tape of P. It can only interact with P as a verifier could do.

Rewindable: Differently from the verifier, K can rewind P, that is, K can at any step of the interaction, come back to a previous step, without starting again from the beginning.

This notion is not so intuitive, however we will discuss again the knowledge extractor after the definition of Σ-protocol. Indeed a knowledge extractor for Σ-protocol is very easy to build and it can be done in general for any Σ-protocol.

Notation

In order to make the text more readable we will use the notation $s \in_R S$ to denote a uniform and random selection of an element s from S. Moreover we will denote R_P and R_V the random tape of P and V and as (P, V) the protocol that P and V will execute.

Algorithms

The protocol (P, V) consist of four polynomial time algorithms

- A takes as inputs x, w and possibly random bits r_a from R_P and outputs the initial message a
- C takes as only input random bits r_c from the random tape R_V and outputs a challenge c.
- Z takes as inputs x, w, a, c, r_a and possibly random bits r_z from R_P and outputs the reply z.
- ϕ takes as input (x, a, c, z) and outputs 0 or 1 ("rejected" or "accepted")

Moreover we will call an accepting conversation the string (x, a, c, z) such that

$$\phi((x, a, c, z)) = 1$$

and a collision a pair of accepting conversations $((x, a, c, z), (x, a, c', z'))$ such that

$$c \neq c' \text{ and } z \neq z'$$

Requirement

As in the case of the proof systems we want to put some requirements on the three main properties of those protocols: completeness, soundness and zero-knowledge. The first two are stronger than in a proof system while the third (that in a proof system is not even requested) is a stronger version of the honest verifier zero-knowledge. The formal description will be included directly in the definition of Σ-protocols.

2.3.2 Definition

We are now ready to give a complete definition of Σ protocols.

Definition 2.3.3 (Σ protocol)**.** *A pair of probabilistic polynomial time interactive machines (P, V) is a Σ-protocol for relation R if they follow the interaction steps:*

1. *P runs $A(x, w, r_a)$ and sends the output a to V through the communication tape*
2. *V runs $C(r_c)$ and sends the output c to P through the communication tape (from now on we will say the V takes $c \in_R \{0,1\}^*$)*
3. *P runs $Z(x, w, c, r_a, r_z)$ and sends the output z to V through the communication tape*
4. *V runs $\phi((x, a, c, z))$ and outputs 0 or 1.*

and the following conditions hold:

Completeness: *If $(x, w) \in R$ then the protocol always accepts it.*

Special Soundness: *If $x \in RX$: given a collision for x there exists a probabilistic polynomial time machine (depending on (P, V) and R) that computes w such that $(x, w) \in R$.*
If $x \notin RX$: there does not exist a collision for x

Special Honest Verifier Zero-Knowledge: *For all $x \in RX$ there exists a probabilistic polynomial time algorithm M which on input x and random challenge c outputs an accepting conversation with the same probability distribution as conversations between P and V.*

Knowledge Extractor for Σ-protocols.

Thanks to the special soundness of the Σ-protocols, a knowledge extractor for w can be easily built as follow: consider an algorithm K on input x and rewindable black-box access to P that follows this steps:

1. K receives a from P
2. K selects a challenge c using the algorithm C and sends it to P

3. K receives the reply z from P
4. K rewinds P to step 2, selects a different challenge c' and sends it to P
5. K receives the (new) reply z'
6. K computes w from the collision $((x, a, c, z), (x, a, c', z'))$ using the probabilistic polynomial time algorithm whose existence is guaranteed by the special soundness property.

This is a polynomial time algorithm that computes a witness for the relation R, hence it is a knowledge extractor.

2.3.3 A relaxed definition

The conditions in the definition of a Σ-protocols are rather strong. In this thesis we will need, for some protocols, a slightly more flexible definition in order to keep the efficiency (jumping ahead the communication cost) of the protocol low enough for our purpose. This definition is different in two aspects:

- We want that the verifier can select an element t at random from a given set, and can send t to P before receiving anything from it.
- We want a relaxed condition of special soundness in the case that, given a relation R and a common input x, x is not in RX, i.e. does not exist a w such that $(x, w) \in R$

To obtain the first aspect we have to introduce a new algorithm T that takes as input a number m of bits from the random tape R_V (call it r_t) and outputs t. Moreover we modify all the algorithms A, C, Z, ϕ in order to let them have also such t as input and we modify the notion of collision: we will say that a collision is a pair of accepting conversation $((x, t, a, c, z), (x, t, a, c', z'))$ such that

$$c \neq c' \text{ and } z \neq z'$$

The formal definition is the following:

Definition 2.3.4 (Public Coin Relaxed Σ-protocol)**.** *A pair of probabilistic polynomial time interactive machines* (P, V) *is a* public-coin relaxed Σ-protocol *for relation* R *if they follow the interaction steps:*

1. *V runs $T(r_t)$ and sends the output t to P through the communication tape*
2. *P runs $A(x, w, r_a, t)$ and sends the output a to V through the communication tape*
3. *V runs $C(r_c, t)$ and sends the output c to P through the communication tape (from now on we will say the V takes $c \in_R \{0,1\}^*$)*
4. *P runs $Z(x, w, c, r_a, r_z, t)$ and sends the output z to V through the communication tape*
5. *V runs $\phi((x, t, a, c, z))$ and outputs 0 or 1.*

and the following conditions hold:

Completeness: *If $(x, w) \in R$ then the protocol always accept it.*

Relaxed Special Soundness: *If $x \in RX$: given a collision for x there exists a probabilistic polynomial time machine (depending on (P, V) and R) that computes w such that $(x, w) \in R$.*
If $x \notin RX$: the probability that a collision exists is negligible in the length of x.

Special Honest Verifier Zero-Knowledge: *For all $x \in RX$ there exists a probabilistic polynomial time algorithm M which on input x and random challenge c outputs an accepting conversation with the same probability distribution as conversations between P and V.*

2.4 Cut and Choose

In this section we want to show a general construction to build Σ-protocols for a particular class of relations. This construction, known as "Cut and Choose", also fits our initial problem (as we will show in chapter 4) and it is the starting point for the improvement that we will show.
To explain informally the idea we recall the example at the beginning of this chapter and we add a possible solution:

Example 2.4.1. There are two rooms (A and B). Every room has its own entrance and there is a door connecting them. This door can be opened

inserting the right secret code. Peggy claims that she knows this code and wants to sell it to Victor. Clearly Peggy doesn't want to reveal the code before receiving the money from Victor but Victor doesn't want to pay Peggy before being sure that the code is correct.
They can proceed as follow:

1. Peggy secretly chooses room A or room B and enters in it.
2. Victor chooses room A or B and asks Peggy to exit from it
3. Peggy exits from the door chosen by Victor (if needed opening the door between the rooms with the secret code)
4. They repeat the protocol for a sufficient number of times, until Victor is convinced that Peggy knows the code.

It should be clear that if Peggy really knows the secret code, she can always exit from the room requested by Victor (Completeness). Moreover if Peggy knows how to exit from both rooms, then she can open the door (or at least go from a room to the other which is equivalent to knowing the secret code) (Special Soundness). Finally the behavior of Peggy can be simulated by just choosing the room first, then entering and exiting from that room. (SHZK)

Bringing this construction to mathematical terms we have the following:

Relation

Consider any equivalence relation that cannot be checked trivially (such as graph isomorphism, or permutation of encrypted vector) and consider the relation R such that $(x, w) \in R$ if x is a pair (A_0, A_1) of representatives of the same class of the equivalence relation, and w is the equivalence φ between these two representatives.

Definition

Protocol 2.4.2 (Cut and Choose)**.** *Consider the relation R as defined before, a pair of PITM (P, V) and the following protocol:*

1. *P chooses $r \in_R \{0, 1\}$ and a new equivalence ϕ at random in the set of all possible equivalences*
   P computes $B = \phi(A_r)$ and sends it to V

2. *V chooses $c \in_R \{0,1\}$ and sends it to P*

3. *P replies $z = \phi$ (in case of $r = c$) or $z = \phi\varphi^{-1}$ in case of $r \neq c$.*

4. *V checks that $z(A_c) = B$*

Theorem 2.4.3. *The protocol 2.4.2 is a Σ-protocol*

Proof. Cleary the protocol 2.4.2 respects the form of a Σ- protocol. We have to check that it respects the three properties:

Completeness: Suppose that $((A_0, A_1), \varphi) \in R$, then by construction $z(A_c) = B$ and hence V will accept the protocol with probability 1.

Special Soundness: Suppose that $((A_0, A_1), \varphi) \in R$ then for $((B, 0, \rho), (B, 1, \rho'))$ one can computes $\varphi = \rho'^{-1}\rho$. Indeed $\rho(A_0) = B = \rho'(A_1)$, hence $\rho'^{-1}\rho(A_0) = A_1$.
Suppose that $((A_0, A_1), \varphi) \notin R$. Then there do not exist two equivalences ϕ and ψ such that $B = \phi(A_c)$ and $B = \psi(A_{1-c})$, hence a collision does not exist.

SHVZK: Supposing that the verifier is honest we can build a simulator $M(A_0, A_1)$ performing:

- M selects $c' \in_R \{0,1\}$.
- M selects ψ at random in the set of all possible equivalences.
- M computes $B' = \psi(A_c)$.
- M outputs (B', c', ψ)

Now the verifier will always accept this conversation since $B' = \psi(A_c)$ by construction. Now we have to prove that (B', c', ψ) has the same probability distribution of (B, c, z). Clearly c and c' have the same probability distribution since they are chosen independently and uniformly at random in$\{0,1\}$. The selection of ψ is independent of c' and uniform in the set of equivalences. Conditioned to c, also the distribution of z is uniformly at random, since, if $r = c$ then $z = \phi$ and ϕ is selected uniformly at random while if $r \neq c$ then φ^{-1} is fixed and hence $\phi\varphi^{-1}$ still has the uniform distribution. Finally also B' and B have the same distribution (conditioned to (c', ψ) and (c, z)) since they are built in the same way: $B' = \psi(A_{c'})$ and $B = z(A_c)$, and (c', ψ),(c, z) have the same independent uniform distribution. □

Chapter 3

Auxiliary protocols

Introduction

In this chapter we explicitly describe some protocols that will be useful in dealing with our initial problem.

Type of protocols

The protocols that we are going to describe are built up to verify relations between committed values. They rely on the homomorphic properties of those schemes and we will implicitly assume the use of Pedersen commitment, even if they work with any homomorphic commitment schemes. Most of the protocols will be Σ-protocol, but this will not always be the case. In general they are all public coin relaxed Σ-protocols (definition 2.3.4).

Zero-knowledge and Commitment Schemes

When using commitment schemes as parts of a zero-knowledge protocol, one must take into account the dependency between the flavor of the commitment schemes properties and the flavor of zero-knowledge. Intuitively if a commitment is computationally hiding there is no hope that a protocol that use it, is perfect zero-knowledge since a super-polynomial adversary can open the commitment and hence he definitely gains knowledge. On the other hand, in the very general definition of proof system, the prover doesn't have bounds on the computational power which means that he can be able to change the

commitment, if the schemes is computationally binding.

In this chapter (and in the following) we will assume that our commitment scheme is unconditionally hiding and computationally binding and that the prover has polynomially bounded computational power. Hence our protocols can "reach" the perfect zero-knowledge and no prover is able to "force" the binding property.

Zero-knowledge (ZK) and SHVZK

In general proving that a protocol is ZK is much more difficult than proving that it is HVZK or SHVZK. The problem of converting a HVZK protocol in a ZK protocol has been studied extensively. It was first studied under the DLP assumption by Bellare, Micali and Ostrovsky in [12] while a generalization to one-way assumption was given by Ostrovsky, Venkatesan and Yung in [13]. In this and the following chapter, we will only prove the SHVZK property. Since standard techniques of converting into ZK have been published this is strong enough for our purposes.

Generator

When dealing with a relation R we have to analyze how the common input x is generated. In fact, if finding a witness w for a given x is computationally easy, then the verifier itself can compute it, and hence the protocol proves a trivial fact.
In all the protocols of this chapter, the relations will involve commitment schemes. Hence the public input will be composed of a public key and at least one committed value.
Such public key can be generated using the generator of the commitment scheme. Thus we will always assume that the key is correctly generated, where correctly means with respect to the properties of the commitment scheme. Moreover we also assume that the binary length of the security parameter ℓ is greater than the binary length of the common input x of the protocol.
In general, also the generation of the committed values must be analyzed but in this thesis we will use only public coin relaxed Σ-protocols (Σ-protocols can be seen as a subclass). All the properties of this class (SHVZK, special soundness and completeness) and the unconditionally hiding property of the

commitment scheme do not depend on the distributions of the committed values. Hence we will not specify how these values are generated.

Algorithms

In chapter 2 P and V were modeled as machines and they accomplish any tasks via proper algorithms, however in this chapter we will describe the protocols as if P and V were human that can compute, send or select values. This type of description, even if less rigorous, helps the reader in the understanding of the protocol's steps without affecting any properties or proofs.

Composition of protocols

All the protocols that we build in this chapter will be used as subprotocols for other protocols. Even if the subprotocols will be described separately, whenever we will say that two protocols are run in parallel we will mean that, instead of running the first protocol and then the second (this would be serial composition) we run the first steps of both protocols, then the second and so on. We also underline that if two protocols are run in parallel, the random choices made by V in every step are always the same for both the subprotocols.

Communication cost

In this chapter we will also analyze the communication cost of the protocols. We will use the O-notation in order to have an asymptotic definition. We will compute such cost choosing as measure unit $M_G(\ell)$ where $M_G(\ell)$ is as in the definition 1.3.8. Even if our measure unit depends on the output of generator of the commitment scheme, the analysis still makes sense. In fact the set-up phase of the commitment scheme can be done once and for all and must not be repeated every time. Hence, we can compare the efficiency of different protocols, assuming that they use the same output of the generator of the commitment scheme.

3.1 Openable Commitment Protocol

This protocol is implicitly contained in the multiplication protocol describe in [4].

Aim

The first property that we want to be able to verify is that the prover can open a commitment. This protocol will be essential in almost all the other protocols described in this thesis. We assume to be using Pedersen commitment.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$ together with the commitment of a value $C \in G$ while the witness will be such value $b \in \mathbb{Z}_q$ and the random element $r_b \in \mathbb{Z}_q$ that hides it. More precisely we say that $(x, w) \in R$ if $x = (pk, C)$ and $w = (r_b, b)$ such that $C = \texttt{commit}_{pk}(r_b, b)$. Clearly if we want to generate an element in this relation we run the generator of the commitment scheme to select a public key $pk = (q, g, h, \texttt{lab})$, then we select b and r_b at random in $\mathbb{Z}_q$ and we compute $C = \texttt{commit}_{pk}(r_b, b)$.

Protocol

Protocol 3.1.1 (Openable commitment protocol)**.**
Consider the relation R as defined before, a pair of PITM (P, V) and the following protocol:

Public Input: (pk, C)

Private Input: (r_b, b)

Interaction:

1. *P selects $u, r_u \in_R \mathbb{Z}_q$*
   P computes
   $$a = \mathrm{Com}(r_u, u),$$
   P sends a to V.

2. *V selects $e \in_R \mathbb{Z}_q$*
   V sends e to P.
3. *P computes*
   $\mu = u + eb,$
   $\rho = r_u + er_b,$
   P sends μ, ρ to V
4. *V checks*
   $\mathrm{Com}(\rho, \mu) \stackrel{?}{=} a \cdot C^e,$

Theorem 3.1.2. *The pair (P, V) as described in protocol 3.1.1 is a Σ-protocol for relation R.*

Proof. Cleary the protocol 3.1.1 respects the form of a Σ- protocol. We have to check that it respects the three properties:

Completeness: Suppose that $((pk, C), (b, r_b)) \in R$

$$\begin{aligned} \mathrm{Com}(\rho, \mu) = &\quad \mathrm{Com}(r_u + er_b, u + eb) &= \\ = &\quad \mathrm{Com}(r_u, u) \cdot \mathrm{Com}(er_b, eb) &= \\ = &\quad a \cdot \mathrm{Com}(r_b, b)^e &= a \cdot C^e \end{aligned}$$

hence V will accept the protocol with probability 1.

Special Soundness: Suppose that $(pk, C) \in RX$ (i.e that exists a w such that $((pk, C), w) \in R$.
Then suppose that we have a collision $(((pk, C), a, e, (\rho, \mu))), ((pk, C), a, e', (\rho', \mu')))$. Observing that $(e - e')$ is invertible in $\mathbb{Z}_q$, we can compute:

$$(\mu - \mu')(e - e')^{-1} = b$$

and

$$(\rho - \rho')(e - e')^{-1} = r_b$$

such that $\mathrm{Com}(r_b, b) = C$ and hence we computed a witness (r_b, b) for C.
Suppose that $(pk, C) \notin RX$. Then, if a collision exists, proceeding as before we can compute a witness for (pk, C). But this is a contradiction since no witness for (pk, C) exists.

SHVZK: Supposing that the verifier is honest we can build a simulator $M((pk, C))$ performing:

- M selects $e' \in_R \mathbb{Z}_q$.
- M selects $\mu', \rho' \in_R \mathbb{Z}_q$
- M computes $a' = \mathrm{Com}(\rho', \mu') \cdot C^{-e'}$
- M outputs $((a', e', (\rho', \mu'))$

Now the verifier will always accept this conversation since:

$$a' \cdot C^{e'} = \mathrm{Com}(\rho', \mu') \cdot C^{-e'} \cdot C^{e'} = \mathrm{Com}(\rho', \mu')$$

by construction. Clearly e (in the real protocol) and e' (in the simulation) are selected independently and uniformly at random. Moreover by the unconditionally hiding of the commitment scheme the authentication message a in the real protocol has uniform distribution over G and this distribution is independent from e. $\mathrm{Com}(\rho', \mu')$ has uniform distribution over G for the same reason and, multiplying it by $C^{-e'}$, this distribution remains uniform over G. Thus a' has uniform distribution over G and such distribution is independent from e'. Finally (ρ, μ) has the uniform distribution over $\mathbb{Z}_q^2$ since u and r_u are selected uniformly at random and the additions of eb and er_b are one to one maps. Moreover (ρ', μ') has the same distributions since both the components are selected independently and uniformly at random in $\mathbb{Z}_q$. Hence the distribution of $((a', e', (\rho', \mu'))$ is the same as in a real conversation. □

Proposition 3.1.3. *The protocol 3.1.1 has a communication cost of* $O(1)$

Proof. (P, V) has to communicate a, e, μ, ρ, hence the communication cost is constant in $M_G(\ell)$. Thus the communication cost is $O(1)$. □

3.2 Multiplication protocol

This protocol is described for the first time in an article of Cramer and Damgård on zero-knowledge proofs for arithmetic circuits over finite prime fields ([4]).

Aim

The aim of the multiplication protocol is to verify, given two commitments B and C, that a third commitment D hides the product of the values hidden by B and C.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$ and the commitments of three values $B, C, D \in G$ while the witness will be two of this values $b, c \in \mathbb{Z}_q$ and the random elements $r_b, r_c, r_d \in \mathbb{Z}_q$ that hides those values and their product. More precisely we say that $(x, w) \in R$ if

$$x = (pk, B, C, D) \text{ and } w = (b, r_b, c, r_c, r_d)$$

such that

$$B = \texttt{commit}_{pk}(r_b, b), \quad C = \texttt{commit}_{pk}(r_c, c), \quad D = \texttt{commit}_{pk}(r_d, bc).$$

Clearly if we want to generate an element in this relation we run the generator of the commitment scheme to select a public key $pk = (q, g, h, \texttt{lab})$, then we select b, c, r_b, r_c, r_d at random in $\mathbb{Z}_q$ and we compute B, C, D.

Protocol

Protocol 3.2.1 (Multiplication protocol)**.**
Consider the relation R as defined before, a pair of PITM (P, V) and the following three protocols run in parallel with the same challenge $e \in_R \mathbb{Z}_q$:

Public input: (pk, B, C, D)

Private input: (b, r_b, c, r_c, r_d)

Interaction:

1. *(P, V) runs the protocol 3.1.1 to check that the prover can open B*
2. *(P, V) runs the protocol 3.1.1 to check that the prover can open C*
3. *(P, V) runs the following protocol to check that D has the right form*

(a) *P selects* $\nu \in_R \mathbb{Z}_q$
P takes the u selected in 1.
P computes
$a = C^u \cdot \mathrm{Com}(\nu, 0)$
P sends a to V.

(b) *V takes e selected in 1.*
V sends e to P.

(c) *P computes*
$\mu = u + eb,$
$v = \nu + e(r_d - r_c b)$
P sends μ, v *to V*

(d) *V checks*
$C^\mu \cdot \mathrm{Com}(v, 0) \stackrel{?}{=} a \cdot D^e$

Theorem 3.2.2. *The pair* (P, V) *as described in protocol 3.2.1 is a* Σ*-protocol for relation* R*.*

Proof. We will not prove again that the first two subprotocols respects the property of the Σ protocol. Clearly the third subprotocol of 3.2.1 respects the form of a Σ- protocol. We have to check that it respects the three properties:

Completeness: Suppose that $((pk, B, C, D), (b, r_b, c, r_c, r_d)) \in R$. Then

$$\begin{aligned} &C^\mu \cdot \mathrm{Com}(v, 0) = \\ &= \mathrm{Com}(r_c, c)^u \cdot \mathrm{Com}(br_c, bc)^e \cdot \mathrm{Com}(\nu, 0) \cdot \mathrm{Com}(r_d - r_c b, 0)^e = \\ &= a \cdot \mathrm{Com}(br_c + r_d - br_c, bc)^e = \\ &= a\,\mathrm{Com}(r_d, bc)^e = a \cdot D^e \end{aligned}$$

Hence V will accept the protocol with probability 1.

Special Soundness: Suppose that $(pk, B, C, D) \in RX$ (i.e that exists a w such that $((pk, B, C, D), w) \in R$.)
Then suppose that we have a collision

$$(((pk, B, C, D), a, e, (v, \mu))), ((pk, B, C, D), a, e', (v', \mu'))).$$

Observing that $(e - e')$ is invertible in $\mathbb{Z}_q$ and that from the special soundness of the protocol 3.1.1 we can obtain b, r_b, c, r_c, we can compute:

$$(v - v')(e - e')^{-1} + r_c b = r_d - r_c b + r_c b = r_d$$

and hence a witness (b, r_b, c, r_c, r_d). Suppose that $(pk, B, C, D) \notin RX$. Then, if a collision exists, proceeding as before we can compute a witness for (pk, B, C, D). But this is a contradiction since such witness does not exist.

SHVZK: Supposing that the verifier is honest and that we have the simulator for the protocol 3.1.1 we can build a simulator $M((pk, B, C, D))$ performing:

- M include the simulator for 3.1.1 and run it twice with selection $e' \in_R \mathbb{Z}_q$,
- M selects $v' \in_R \mathbb{Z}_q$ and takes μ' and e' as selected in the previous step.
- M computes $a' = C^{\mu'} \cdot \mathrm{Com}(v', 0) \cdot D^{-e'}$
- M outputs $((a', e', (v', \mu'))$

Now the verifier will always accept this conversation since:

$$a \cdot D^e = C^{\mu} \cdot \mathrm{Com}(v, 0) \cdot D^{-e} \cdot D^e = C^{\mu} \cdot \mathrm{Com}(v, 0)$$

by construction. Obviously the first two steps are correctly simulated by the proof of protocol 3.1.1. Now consider the subprotocol in the third step: clearly e (in the real protocol) and e' (in the simulation) are selected independently and uniformly at random. Moreover by the unconditionally hiding of the commitment scheme the authentication message a in the third step of the real protocol has uniform distribution over G and this distribution is independent from e. $\mathrm{Com}(v', 0)$ has uniform distribution over G for the same reason and, multiplying it by $C^{\mu'}$ and $D^{-e'}$, this distribution remains uniform over G. Thus a' has uniform distribution over G and such distribution is independent from e'. Finally (v, μ) has the uniform distribution over $\mathbb{Z}_q$ since u and ν are selected uniformly at random and the additions of eb and $e(r_d - r_c b)$ are one to one maps. Moreover (v', μ') has the same distributions since both the components are selected independently and uniformly at random in $\mathbb{Z}_q$. Hence the distribution of $((a', e', (v', \mu'))$ is the same as in a real conversation. □

Proposition 3.2.3. *The protocol 3.2.1 has a communication cost of* $O(1)$

Proof. (P, V) uses twice a protocol of cost $O(1)$ for the first two subprotocols, for the third (P, V) has to communicate (a, e, μ, v), hence the communication cost of the subprotocol is constant in $M_G(\ell)$. Hence the total communication cost is still $O(1)$. □

3.3 Inner Product Protocol

Aim

This protocol is a generalization of the previous one. Instead of verifying that a commitment hides the product of two committed values we want to verify that given two vectors of commitments $B = (B_1, \ldots, B_n)$ and $C = (C_1, \ldots, C_n)$, a commitment D hides the inner product of the vector $b = (b_1, \ldots, b_n)$ and $c = (c_1, \ldots, c_n)$ which components are the values hidden by the corresponding components of B and C.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$, two n-vector of commitment $B, C \in G \times \ldots \times G$ and a single commitment $D \in G$ while the witness will be the two vectors of values $b, c \in \mathbb{Z}_q^n$, the n-vectors of random elements $r_b, r_c \in \mathbb{Z}_q^n$ that hide those values and the random element $r_d \in \mathbb{Z}_q$ that hides their inner product. More precisely we say that $(x, w) \in R$ if

$$x = (pk, B, C, D) \text{ and } w = (b, r_b, c, r_c, r_d)$$

such that for all $i \in \{1, \ldots, n\}$

$$B_i = \texttt{commit}_{pk}(r_{b_i}, b_i), \quad C_i = \texttt{commit}_{pk}(r_{c_i}, c_i), \quad D = \texttt{commit}_{pk}(r_d, \sum_{i=1}^{n} b_i c_i).$$

Clearly if we want to generate an element in this relation we run the generator of the commitment scheme to select a public key $pk = (q, g, h, \texttt{lab})$, then we select for all $i \in \{1, \ldots, n\}$ $b_i, c_i, r_{b_i}, r_{c_i}, r_d$ at random in $\mathbb{Z}_q$ and we compute B, C, D.

Protocol

Protocol 3.3.1 (Inner Product Protocol)**.**

Consider the relation R as defined before, a pair of PITM (P, V) and the following protocols run in parallel with the same challenge $e \in_R \mathbb{Z}_q$:

Public input: (pk, B, C, D)

Private input: (b, r_b, c, r_c, r_d)

Interaction:

1. *For all $i \in \{1, \ldots, n\}$ (P, V) runs the protocol 3.1.1 to check that the prover can open B_i.*
2. *For all $i \in \{1, \ldots, n\}$ (P, V) runs the protocol 3.1.1 to check that the prover can open C_i with challenge e as selected in 1.*
3. *(P, V) runs the following protocol to check that D has the right form*
   (a) *P selects $\nu \in_R \mathbb{Z}_q$*
   P takes, for all i, the u_i selected in the i^{th} iteration in step 1.
   P computes
   $a = \prod_{i=1}^{n} C_i^{u_i} \cdot \mathrm{Com}(\nu, 0)$
   P sends a to V.
   (b) *V takes e selected in 1.*
   V sends e to P.
   (c) *P computes*
   For all i, $\mu_i = u_i + eb_i$,
   $v = \nu + e(r_d - \sum_{i=1}^{n} r_{c_i} b_i)$
   P sends for all i, μ_i, v to V
   (d) *V checks*
   $\prod_{i=1}^{n} C_i^{\mu_i} \cdot \mathrm{Com}(v, 0) \stackrel{?}{=} a \cdot D^e$

Theorem 3.3.2. *The pair (P, V) as described in protocol 3.3.1 is a Σ-protocol for relation R.*

Proof. Completeness: Suppose that $((pk, B, C, D), (b, r_b, c, r_c, r_d)) \in R$. Then

$$\begin{aligned}
&\prod_i C_i^{\mu_i} \cdot \mathrm{Com}(v, 0) = \\
&= \prod_i (\mathrm{Com}(r_{c_i}, c_i)^{u_i} \cdot \mathrm{Com}(b_i r_{c_i}, b_i c_i)^e) \cdot \mathrm{Com}(\nu, 0) \cdot \\
&\qquad \cdot \mathrm{Com}(r_d - \sum_i r_{c_i} b_i, 0)^e = \\
&= a \cdot \mathrm{Com}(\sum_i b_i r_{c_i} + r_d - \sum_i b_i r_{c_i}, \sum_i b_i c_i)^e = \\
&= a \cdot \mathrm{Com}(r_d, \sum_i b_i c_i)^e = a \cdot D^e
\end{aligned}$$

Hence V will accept the protocol with probability 1.

Special Soundness: Suppose that $(pk, B, C, D) \in RX$ (i.e that exists a w such that $((pk, B, C, D), w) \in R$.)
Then suppose that we have a collision

$$(((pk, B, C, D), a, e, (v, \mu_1, \ldots, \mu_n))), ((pk, B, C, D), a, e', (v', \mu_1', \ldots, \mu_n))).$$

Observing that $(e - e')$ is invertible in $\mathbb{Z}_q$ and that from the special soundness of the protocol 3.1.1 we can obtain b, r_b, c, r_c, we can compute:

$$(v - v')(e - e')^{-1} + \sum_i r_{c_i} b_i = r_d - \sum_i r_{c_i} b_i + r_{c_i} b_i = r_d$$

and hence a witness (b, r_b, c, r_c, r_d). Suppose that $(pk, B, C, D) \notin RX$. Then, if a collision exists, proceeding as before we can compute a witness for (pk, B, C, D). But this is a contradiction since such witness does not exist.

SHVZK: Supposing that the verifier is honest and that we have the simulator for the protocol 3.1.1 we can build a simulator $M((pk, B, C, D))$ performing:

- M include the simulator for 3.1.1 and run it $2n$ times with selection $e' \in_R \mathbb{Z}_q$,

- M selects $v' \in_R \mathbb{Z}_q$ and takes for all i, μ'_i and e' as selected in the previous step in the i^{th} iteration.
- M computes $a' = \prod_i C_i^{\mu'_i} \cdot \mathrm{Com}(0, v') \cdot D^{-e'}$
- M outputs $((a', e', (v', \mu'_1, \ldots, \mu'_n))$

Now the verifier will always accept this conversation since:

$$a \cdot D^e = \prod_i C_i^{\mu_i} \cdot \mathrm{Com}(v, 0) \cdot D^{-e} \cdot D^e = \prod_i C_i^{\mu_i} \cdot \mathrm{Com}(v, 0)$$

by construction. Obviously the first $2n$ steps are correctly simulated by the proof of protocol 3.1.1. Now consider the subprotocol in the third step: clearly e (in the real protocol) and e' (in the simulation) are selected independently and uniformly at random. Moreover by the unconditionally hiding property of the commitment scheme the authentication message a in the third step of the real protocol has uniform distribution over G and this distribution is independent from e. $\mathrm{Com}(v', 0)$ has uniform distribution over G for the same reason and, multiplying it by $\prod_i C^{\mu'_i}$ and $D^{-e'}$, this distribution remains uniform over G. Thus a' has uniform distribution over G and such distribution is independent from e'. Finally $(v, \mu_1, \ldots, \mu_n)$ has the uniform distribution over $\mathbb{Z}_q^{n+1}$ since for all i, u_i and ν are selected uniformly at random and the additions of eb_i and $e(r_d - \sum r_{c_i} b_i)$ are one to one maps. Moreover $(v', \mu'_1, \ldots, \mu'_n)$ has the same distributions since all the components are selected independently and uniformly at random in $\mathbb{Z}_q$. Hence the distribution of $((a', e', (v', \mu'_1, \ldots, \mu'_n))$ is the same as in a real conversation.

□

Proposition 3.3.3. *The protocol 3.3.1 has a communication cost of $O(n)$*

Proof. (P, V) uses $2n$ times a protocol of cost $O(1)$. The last subprotocol has a communication cost of $O(1)$ just as in the previous protocol. Hence the total communication cost is $O(2n + 1) = O(n)$. □

3.4 Matrix Multiplication Protocol

Aim

The protocol 3.3.1 can be used to prove that the product of two $n \times n$ matrices B and C of committed values, hides the same values as a third

matrix D. This can be done just using the protocol 3.3.1 n^2 times, one for every inner product between a row of B and a column of C. However this would have a communication cost of $O(n^3)$. In this section we will describe a different protocol with communication cost of $O(n^2)$. This improvement will be possible using the definition of public coin relaxed Σ-protocol (definition 2.3.4). The trick is to check that $BCx = Dx$ where x is a vector of length n uniformly selected over $\mathbb{Z}_q^n$ instead of $BC = D$.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$, two $n \times n$ matrices of commitment $B, C \in \mathcal{M}(n, G)$ and a third $n \times n$ matrix of commitment $D \in \mathcal{M}(n, G)$ while the witness will be the two matrices of values $b, c \in \mathcal{M}(n, \mathbb{Z}_q)$, the matrices of random elements $r_b, r_c \in \mathcal{M}(n, \mathbb{Z}_q)$ that hide those values and the matrix of random elements $r_d \in \mathcal{M}(n, \mathbb{Z}_q)$ that hides the product of b and c. More precisely we say that $(x, w) \in R$ if

$$x = (pk, B, C, D) \text{ and } w = (b, r_b, c, r_c, r_d)$$

such that for all $i \in \{1, \ldots, n\}$ and for all $j \in \{1, \ldots, n\}$

$$B_{ij} = \texttt{commit}_{pk}(r_{b_{ij}}, b_{ij}), \quad C_{ij} = \texttt{commit}_{pk}(r_{c_{ij}}, c_{ij}),$$

$$D_{ij} = \texttt{commit}_{pk}(r_{d_{ij}}, \sum_{k=1}^{n} b_{ik} c_{kj}).$$

Clearly if we want to generate an element in this relation we run the generator of the commitment scheme to select a public key $pk = (q, g, h, \texttt{lab})$, then we select for all $i, j \in \{1, \ldots, n\} b_{ij}, c_{ij}, r_{b_{ij}}, r_{c_{ij}}, r_{d_{ij}}$ at random in $\mathbb{Z}_q$ and we compute B, C, D.

Protocol

Protocol 3.4.1 (Matrix Multiplication Protocol)**.**
Consider the relation R as defined before, a pair of PITM (P, V) and the following protocols run in parallel with the same challenge $e \in_R \mathbb{Z}_q$:

Public input: (pk, B, C, D)

Private input: (b, r_b, c, r_c, r_d)

Interaction:

1. *For all* $i, j \in \{1, \ldots, n\}$ (P, V) *runs the protocol 3.1.1 to check that the prover can open* B_{ij}
2. *For all* $i, j \in \{1, \ldots, n\}$ (P, V) *runs the protocol 3.1.1 to check that the prover can open* C_{ij}
3. *For all* $i, j \in \{1, \ldots, n\}$ (P, V) *runs the protocol 3.1.1 to check that the prover can open* D_{ij}
4. V *selects* $x \in_R \mathbb{Z}_q^n$
   V *sends* x *to* P
   V *computes for all* i
   $\bar{y}_i = \prod_{j=1}^{n} C_{ij}^{x_j}$
   $\bar{z}_i = \prod_{j=1}^{n} D_{ij}^{x_j}$
   P *computes for all* i
   $y_i = \sum_{j=1}^{n} c_{ij} \cdot x_j$
   $r_{y_i} = \sum_{j=1}^{n} r_{c_{ij}} \cdot x_j$
   $z_i = \sum_{j=1}^{n} ((\sum_{k=1}^{n} b_{ik} c_{kj}) \cdot x_j)$
   $r_{z_i} = \sum_{j=1}^{n} r_{d_{ij}} \cdot x_j$
5. *For all* $i \in \{1, \ldots, n\}$ (P, V) *runs the protocol 3.3.1 on common input* $(B_i, \bar{y}, \bar{z}_i)$ *and private input* $(b_i, r_{b_i}, y_i, r_{y_i}, r_{z_i})$*, where* B_i *and* b_i *denotes the* i^{th} *row respectively of* B *and* b*.*

Theorem 3.4.2. *The pair* (P, V) *as described in protocol 3.4.1 is a public-coin relaxed* Σ*-protocol for relation* R*.*

Proof.

Completeness: Suppose that $((pk, B, C, D), (b, r_b, c, r_c, r_d)) \in R$. Then by the completeness of protocol 3.1.1 we have that the steps 1 and 2 are accepted. Moreover by the completeness of protocol 3.3.1 we have that step 4 is always accepted since

$$BC = D \Rightarrow BCx = Dx$$

for all $x \in \mathbb{Z}_q^n$.

Relaxed Special Soundness: Suppose that $(pk, B, C, D) \in RX$ (i.e that exists a w such that $((pk, B, C, D), w) \in R$.)
Then suppose we have a collision. From the special soundness of protocol 3.1.1 we can compute b, r_b, c, r_c, r_d such that

$$((pk, B, C, D), (b, r_b, c, r_c, r_d)) \in R.$$

Now suppose that $(pk, B, C, D) \notin RX$. Define $d \in \mathcal{M}(n, \mathbb{Z}_q)$ as the matrix of values committed in D. Then a collision exists if and only if $x \in \ker(bc - d)$. Indeed consider the reply of the protocol. It contains the $3n^2$ replies of the first $3n^2$ subprotocols and the n replies of the last n subprotocols. The firsts cannot be different by the special soundness of protocol 3.1.1. If $x \notin \ker(bc - d)$ then also the seconds cannot change and a collision does not exists. However if $x \in \ker(bc - d)$ we can have two different accepted conversation since the common input of the subprotocol are in its relation.

The probability over a uniform random choose of x that $x \in \ker(bc-d)$ is $1/q$. Indeed we have at most $\dim_{\mathbb{Z}_q} \ker(bc-d) = n-1$ and hence the firsts $n-1$ components of x can assume any value and the the last has to be a particular value in $\mathbb{Z}_q$. Clearly the probability of having such value is $1/q$. Hence we have that the probability to have a collision is negligible in the length of ℓ (we recall that ℓ is the security parameter of the commitment scheme) and hence also in (pk, B, C, D).

SHVZK: The first two steps can be simulated using the simulator of the protocol 3.1.1, the third can be simulated just selecting a random vector in the same way x is selected in the real protocol and the last step can be simulated using the simulator of the protocol 3.3.1.

□

Proposition 3.4.3. *The protocol 3.4.1 has a communication cost of* $O(n^2)$

Proof. (P, V) uses $2n^2$ times the protocol 3.1.1 which communication cost is $O(1)$, then it sends $x \in \mathbb{Z}_q^n$ (with cost $O(n)$) and finally it repeats n times the protocol 3.3.1 which communication cost is $O(n)$. Thus the total communication cost is $O(n^2)$. □

3.5 Basic 0/1 Protocol

This protocol were firstly described as a general construction that could be applied to any relation R to prove that either $(x_0, w) \in R$ or $(x_1, w) \in R$, without revealing which is the case. ([20]). Here we describe the protocol in a specific case.

Aim

The aim of this protocol is to verify that, given two commitment c_0 and c_1 ones hides a 1 and the other a 0. Clearly we do not want that the prover must reveal which hides the 1 (otherwise he can just open the commitments).

Relation

The public input will be the public key $pk \in \mathcal{PK}$ of the commitment scheme, together with two committed value $c_0, c_1 \in \mathbb{Z}_q$. The witness will be the position $b \in \{0, 1\}$ of the 1 and the two random elements $r_0, r_1 \in \mathbb{Z}_q$ that hide 1 and 0. More precisely we say that $(x, w) \in R$ if $x = (pk, c_0, c_1)$ and $w = (b, r_0, r_1)$ such that $c_b = \texttt{commit}_{pk}(r_b, 1)$ and, define $\bar{b} = 1 - b$, $c_{\bar{b}} = \texttt{commit}_{pk}(r_{\bar{b}}, 0)$

Protocol

Protocol 3.5.1 (0/1 protocol)**.** *Consider the relation R as defined before, a pair of PITM (P, V) and the following protocol:*

Public Input: (pk, c_0, c_1)

Private Input: (b, r_0, r_1)

Interaction:

1. *P selects $t_0, t_1, e_{\bar{b}}, z_{\bar{b}0}, z_{\bar{b}1} \in_R \mathbb{Z}_q$*
   P computes

$$a_{b0} = \mathrm{Com}(t_0, 0)$$
$$a_{b1} = \mathrm{Com}(t_1, 0)$$
$$a_{\bar{b}\bar{b}} = \mathrm{Com}(z_{\bar{b}\bar{b}}, e_{\bar{b}}) c_{\bar{b}}^{-e_{\bar{b}}}$$
$$a_{\bar{b}b} = \mathrm{Com}(z_{\bar{b}b}, 0) c_{b}^{-e_{\bar{b}}}$$

P *sends* $(a_{00}, a_{11}, a_{01}, a_{10})$ *to* V

2. V *selects* $e \in \mathbb{Z}_p$
   V *sends* e *to* P
3. P *computes*

   $e_b = e - e_{\bar{b}}$

   $z_{b0} = t_0 + e_b r_0$

   $z_{b1} = t_1 + e_b r_1$

   P *sends* $(e_0, e_1, z_{00}, z_{01}, z_{10}, z_{11})$ *to* V
4. V *checks if*

   $\mathrm{Com}(z_{00}, e_0) \stackrel{?}{=} a_{00} c_0^{e_0}$

   $\mathrm{Com}(z_{01}, 0) \stackrel{?}{=} a_{01} c_1^{e_0}$

   $\mathrm{Com}(z_{10}, 0) \stackrel{?}{=} a_{10} c_0^{e_1}$

   $\mathrm{Com}(z_{11}, e_1) \stackrel{?}{=} a_{11} c_1^{e_1}$

Theorem 3.5.2. *The pair* (P, V) *as described in protocol 3.5.1 is a* Σ*-protocol for relation* R*.*

Proof.

Completeness: Suppose that $((pk, c_0, c_1), (b, r_0, r_1)) \in R$. Then the completeness come from the fact that $a_{\bar{b}\bar{b}}$ and $a_{\bar{b}b}$ are build to pass the checks while:

$$\begin{aligned} \mathrm{Com}(z_{bb}, e_b) = & \quad \mathrm{Com}(t_b + e_b r_b, e_b) & = \\ = & \quad \mathrm{Com}(t_b, 0) \cdot \mathrm{Com}(r_b, 1)^{e_b} & = a_{bb} c_b^{e_b} \end{aligned}$$

and

$$\begin{aligned} \mathrm{Com}(z_{b\bar{b}}, 0) = & \quad \mathrm{Com}(t_{\bar{b}} + e_b r_{\bar{b}}, 0) & = \\ = & \quad \mathrm{Com}(t_{\bar{b}}, 0) \cdot \mathrm{Com}(r_{\bar{b}}, 0)^{e_b} & = a_{b\bar{b}} c_{\bar{b}}^{e_b} \end{aligned}$$

hence V will always accept the protocol.

Special Soundness: Let $(pk, c_0, c_1) \in RX$ and let

$$\begin{aligned} &(((a_{00}, a_{11}, a_{01}, a_{10}), e, (e_0, e_1, z_{00}, z_{01}, z_{10}, z_{11})), \\ &((a_{00}, a_{11}, a_{01}, a_{10}), e', (e'_0, e'_1, z'_{00}, z'_{01}, z'_{10}, z'_{11}))) \end{aligned}$$

be a collision. Then there is an $i \in \{0,1\}$ such that $e_i \neq e'_i$. Hence

$$\begin{aligned}
\mathrm{Com}((z_{ii} - z'_{ii})(e_i - e'_i)^{-1}, 1) = &\ \mathrm{Com}(z_{ii} - z'_{ii}, e_i - e'_i)^{(e_i - e'_i)^{-1}} = \\
= &\ \left(\frac{\mathrm{Com}(z_{ii}, e_i)}{\mathrm{Com}(z'_{ii}, e'_i)}\right)^{(e_i - e'_i)^{-1}} = \\
= &\ \left(\frac{a_{ii} c_i^{e_i}}{a_{ii} c_i^{e'_i}}\right)^{(e_i - e'_i)^{-1}} = \\
= &\ c_i^{(e_i - e'_i)(e_i - e'_i)^{-1}} = c_i
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{Com}((z_{i\bar{i}} - z'_{i\bar{i}})(e_i - e'_i)^{-1}, 0) = &\ \mathrm{Com}(z_{i\bar{i}} - z'_{i\bar{i}}, 0)^{(e_i - e'_i)^{-1}} = \\
= &\ \left(\frac{\mathrm{Com}(z_{i\bar{i}}, 0)}{\mathrm{Com}(z'_{i\bar{i}}, 0)}\right)^{(e_i - e'_i)^{-1}} = \\
= &\ \left(\frac{a_{i\bar{i}} c_{\bar{i}}^{e_i}}{a_{i\bar{i}} c_{\bar{i}}^{e'_i}}\right)^{(e_i - e'_i)^{-1}} = \\
= &\ c_{\bar{i}}^{(e_i - e'_i)(e_i - e'_i)^{-1}} = c_{\bar{i}}
\end{aligned}$$

Thus $b = i, r_b = (z_{ii} - z'_{ii})(e_i - e'_i)^{-1}$ and $r_{\bar{b}} = (z_{i\bar{i}} - z'_{i\bar{i}})(e_i - e'_i)^{-1}$

SHVZK: Supposing that the verifier is honest we can build a simulator $M((pk, c_0, c_1))$ performing:

- M selects $e' \in_R \mathbb{Z}_q$
- M selects $e'_0, z'_{00}, z'_{01}, z'_{10}, z'_{11} \in_R \mathbb{Z}_q$
- M computes
  $e'_1 = e' - e'_0$
  $a'_{00} = \mathrm{Com}(z'_{00}, e'_0) c_0^{-e'_0}$
  $a'_{01} = \mathrm{Com}(z'_{01}, 0) c_1^{-e'_0}$
  $a'_{10} = \mathrm{Com}(z'_{10}, 0) c_0^{-e'_1}$
  $a'_{11} = \mathrm{Com}(z'_{11}, e'_1) c_1^{-e'_1}$
- M outputs $((a'_{00}, a'_{01}, a'_{10}, a'_{11}), e', (e'_0, e'_1, z'_{00}, z'_{01}, z'_{10}, z'_{11}))$

Such conversation would always be accepted since $(a'_{00}, a'_{01}, a'_{10}, a'_{11})$ are built to pass the checks. Moreover $a'_{\bar{b}\bar{b}}, a'_{\bar{b}b}, e', z'_{\bar{b}\bar{b}}, z'_{\bar{b}b}$ are chosen uniformly at random, just as $a_{\bar{b}\bar{b}}, a_{\bar{b}b}, e, z_{\bar{b}\bar{b}}, z_{\bar{b}b}$ in the real protocol, so this

part of the simulation has the same distribution of a real conversation. Now consider $(a_{b\bar{b}}, a_{bb}, z_{b\bar{b}}, z_{bb})$ and $(a'_{b\bar{b}}, a'_{bb}, z'_{b\bar{b}}, z'_{bb})$. z'_{bb} and $z'_{b\bar{b}}$ have the uniform distribution over Z_q. Also z_{bb} and $z_{b\bar{b}}$ have the uniform distribution over $\mathbb{Z}_q$ since t_0 and t_1 are selected uniformly at random and the additions of er_0 and er_1 are one to one maps. a_{bb} and $a_{b\bar{b}}$ have the uniform distribution (and are independent) by the unconditionally hiding of the commitment scheme. The same for $\mathrm{Com}(z'_{bb}, e'_b)$ and $\mathrm{Com}(z'_{b\bar{b}}, 0)$, and since the multiplication by $c_{\bar{b}}^{-e'_b}$ and $c_b^{-e'_b}$ are one to one maps, also a'_{bb} and $a'_{b\bar{b}}$ have the uniform distribution. Hence the output of M has the same probability distribution of a real accepting conversation.

□

Proposition 3.5.3. *The protocol 3.5.1 has a communication cost of* $O(1)$

Proof. (P, V) has to communicate

$$(a_{00}, a_{11}, a_{01}, a_{10}, e, e_0, e_1, z_{00}, z_{01}, z_{10}, z_{11})$$

hence the communication cost is constant in $M_G(\ell)$. Thus the communication cost is $O(1)$. □

3.6 Generalized 0/1 Protocol

Aim

From the previous protocol we want a generalization of it that allows to prove that, given a vector of commitment $c = (c_1, \ldots, c_n)$, one of the values committed is 1 and all the others are 0. The main idea is to compute for all $i \in \{1, \ldots, n\}$ the pairs $(c_i, \prod_{j \neq i} c_j)$ and use n times in parallel the basic 0/1 protocol. Indeed if c_j hides a value different from 0 or 1, the 0/1 protocol fails in the round j. If the vector is all of 0, the 0/1 protocol fails in every rounds. If there is a 1 in c_i and in c_j with $i \neq j$, then the 0/1 protocol fails in the round i and in the round j. We will assume that n is smaller than the cardinality of the group G choosen by the generator of the commitment scheme. This implies that more than p components equal to 1 can not be hidden in the vector c.

Relation

The public input will be the public key $pk \in \mathcal{PK}$ of the commitment scheme and a vector $c \in \mathbb{Z}_q^n$ of committed value. The witness will be the position $k \in \{1, \dots n\}$ of the 1 and the vector of random elements $r \in \mathbb{Z}_q^n$ that hides the 1 and the 0s. More precisely we said that $(x, w) \in R$ if $x = (pk, c)$ and $w = (k, r)$ such that $c_k = \texttt{commit}_{pk}(r_k, 1)$ and for all $i \neq k$, $c_i = \texttt{commit}_{pk}(r_i, 0)$

Protocol

Protocol 3.6.1 (Generalized 0/1 Protocol)**.** *Consider the relation R as defined before, a pair of PITM (P, V) and the following protocol:*

Public Input: (pk, c)

Private Input: (k, r)

Interaction:

1. *P computes for all $i \in \{1, \dots, n\}$*
   $r'_i = \sum_{j \neq i} r_j$
   P computes for all $i \in \{1, \dots, n\}$
   $c'_i = \prod_{j \neq i} c_j$
2. *For all $i \in \{1 \dots, n\}$, P and V run the protocol 3.5.1 on public input (pk, c_i, c'_i)*

Theorem 3.6.2. *The pair (P, V) as described in protocol 3.6.1 is a Σ-protocol for relation R.*

Proof.

Completeness: If $((pk, c), (k, r)) \in R$ then $\forall i \in \{1, \dots, n\}$, $c_i = \text{Com}(r_i, 1)$, $c_j = \text{Com}(r'_i, 0)$ or $c_i = \text{Com}(r_i, 0)$, $c_j = \text{Com}(r'_i, 1)$. This is because $n < q$, hence there can't be more than q ones, hence the cyclic structure of $\mathbb{Z}_q$ cannot be used to cheat. Thus from the completeness of the basic 0/1 protocol (Protocol 3.5.1) we have that the proof is never rejected.

Special Soundness: Suppose that $(pk, c) \in RX$ and suppose that we have a collision, then by the special soundness of the protocol 3.5.1 we can compute for all $i \in \{1, \dots, n\}$, r_i and b_i. Then $r = (r_1, \dots, r_n)$ and k

is equal to the b_i such that the 1 was not hidden in the sum. Hence we can compute a witness from a collision.
Suppose that $(pk, c) \notin RX$ then from the collision we can compute a witness by the special soundness of protocol 3.5.1, but such witness does not exists, hence also the collision does not exists.

SHVZK: The simulation can be done just using the simulator of the basic 0/1 protocol (Protocol 3.5.1).

□

Proposition 3.6.3. *The protocol 3.6.1 has a communication cost of* $O(n)$

Proof. In terms of communication (P, V) is simply using the protocol 3.5.1 n times, hence the communication cost is $O(n)$ □

3.7 Protocols over $\mathbb{F}_{q^k}$

In all the previous protocols and in the following, the values to commit are chosen in $\mathbb{Z}_q$. Since q is prime, $\mathbb{Z}_q$ is a finite field of q elements and from now on we will denote it with $\mathbb{F}_q$. Later in this thesis we will need to take our values not just in $\mathbb{F}_q$ but also in finite fields of the same characteristic, namely in $\mathbb{F}_{q^k}$ for some $k \in \mathbb{N}$. To generalize the commitment scheme and the protocols, we present in this section the notion of *multiplicative friendly embedding*, we explicitly define one of it and we describe how to use it on commitment and protocols.

Definition 3.7.1 (Multiplicative Friendly Embedding)**.** *The triplet:*

- $n \in \mathbb{N}$
- $\xi : \mathbb{F}_{q^k} \to \mathbb{F}_q^n$
- $\delta : \mathbb{F}_q^n \to \mathbb{F}_{q^k}$

is a multiplicative friendly embedding if ξ *and* δ *are* $\mathbb{F}_q$*-linear,* ξ *is injective and for all* $a, b \in \mathbb{F}_{q^k}$ *it holds:*

$$a \cdot b = \delta(\xi(a) * \xi(b))$$

where $\cdot$ *is usual multiplication in* $\mathbb{F}_{q^k}$ *and* $*$ *is the usual component-wise multiplication of* $\mathbb{F}_q^n$

Construction of a multiplicative friendly embedding

In this section we will construct a multiplicative friendly embedding based on interpolations and evaluations of polynomials. All the theorems will implicitly require that $q \geq 2k-1$. However they do not require that q is a prime, hence if $q > 2$ (as it happens in the case of commitments), we can iterate the construction.

Theorem 3.7.2. *Let $\mathbb{F}_q$ be a field of q elements. Fix an algebraic closure $\overline{\mathbb{F}}_q$ of $\mathbb{F}_q$. Suppose that $x_1, \ldots, x_m \in \overline{\mathbb{F}}_q$ satisfy the following: for each pair (x_i, x_j) with $i \neq j$ the element x_j is not a Galois-conjugate of x_i over $\mathbb{F}_q$. Then the map:*

$$\begin{aligned} \mathcal{E}: \quad \mathbb{F}_q[X]_{\leq M-1} &\to \bigoplus_{i=1}^{m} \mathbb{F}_q(x_i) \\ f &\to (f(x_1), \ldots, f(x_n)) \end{aligned}$$

is an isomorphism of $\mathbb{F}_q$-vector spaces, where

$$M = \sum_{i=1}^{m} \dim_{\mathbb{F}_q} \mathbb{F}_q(x_i)$$

Proof. The proof is a generalization of Lagrange interpolation.

Suppose first $m = 1$.
If $x_1 \in \mathbb{F}_q$, $\mathbb{F}_q(x_1) = \mathbb{F}_q$ and $M = 1$, thus the claim is trivial.
If $x_1 \notin \mathbb{F}_q$, consider the basis $1, \ldots, x_1^{M-1}$ of $\mathbb{F}_q(x_1)$ as $\mathbb{F}_q$-vector space. For $y \in \mathbb{F}_q(x_1)$ we have $\mathcal{E}(f) = y$, where $f(X)$ is the (unique) polynomial of degree at most $M-1$ whose coefficient are the coordinate of y according to the previous basis.

Now suppose $m > 1$.
For all $i \in \{1, \ldots, m\}$ define:

- The monic polynomial in $\mathbb{F}_q[X]$ of minimal degree having x_i as a zero. Denotes it by $h_i(X)$ and observe that $\deg(h_i) = \dim_{\mathbb{F}_q} \mathbb{F}_q(x_i)$.
- Define
$$\delta_i = \prod_{1 \leq k \leq m, k \neq i} h_i(X)$$
and observe that $\delta_i(x_j) = 0$ if $i \neq j$ and that $\deg(\delta_i) = M - \dim_{\mathbb{F}_q} \mathbb{F}_q(x_i)$.

- Define $z_i = \delta_i(x_i)$ and observe that by the assumption on the x_i, $z_i \in \mathbb{F}_q(x_i) \setminus \{0\}$.

$\mathcal{E}$ is surjective, indeed:
let $y \in \bigoplus_{i=1}^{m} \mathbb{F}_q(x_i)$, for all $i \in \{1, \ldots, m\}$ choose $f_i(x) \in \mathbb{F}_q[X]$ such that $\deg(f_i) \leq \dim_{\mathbb{F}_q} \mathbb{F}_q(x_i)$ and $f_i(x_i) = y_i z_i^{-1}$ (such f_i exists by the claim in the case $m = 1$).
Then define

$$f(X) = \sum_{i=1}^{m} \delta_i(X) f_i(X) \in \mathbb{F}_q[X]$$

We have:

$$\deg(f) \leq \max_i((M - \dim_{\mathbb{F}_q} \mathbb{F}_q(x_i)) + (\dim_{\mathbb{F}_q} \mathbb{F}_q(x_i) - 1)) = M - 1$$

and for all $j \in \{1, \ldots, m\}$

$$f(x_j) = \sum_{i=1}^{m} \delta_i(x_j) f_i(x_j) = \delta_j(x_j) f_j(x_j) = z_j \frac{y_j}{z_j} = y_j$$

Hence $f(X) \in \mathbb{F}_q[X]_{\leq M-1}$ and $\mathcal{E}(f) = y$.
$\mathcal{E}$ is injective, indeed:
Suppose $\mathcal{E}(f) = 0$. It implies $f(x_j) = 0$ for all $j \in \{1, \leq, m\}$. Hence $f(X)$ is divisible by each of the $h_i(X)$, which, by the condition on the x_j are coprime. Therefore $f(X)$ is divisible by the product of all the $h_i(x)$ which has degree M. Since $\deg(f) \leq M - 1$ $f(X)$ is the zero polynomial. □

The idea is now to build an multiplicative friendly embedding $(2k-1, \phi, \psi)$, using the fact that by theorem 3.7.2, $\mathbb{F}_{q^k} \cong \mathbb{F}_q[X]_{\leq k-1}$ and using the evaluation map in $2k-1$ distinct elements of $\mathbb{F}_q$

Definition 3.7.3. *Let $x_0 \in \overline{\mathbb{F}}_q$ be such that $\mathbb{F}_q(x_0) = \mathbb{F}_{q^k}$ and $x_1, \ldots, x_{2k-1} \in \mathbb{F}_q$ be such that for all i, $x_i \neq x_j$ for all $j \neq i$. Then we define the map ϕ as:*

$$\begin{aligned} \phi : \quad \mathbb{F}_{q^k} &\to \mathbb{F}_q^{2k-1} \\ a &\to (f(x_1), \ldots, f(x_{2k-1})) \end{aligned}$$

where f is the unique polynomial of degree at most $k-1$ such that $f(x_0) = a$.

Definition 3.7.4. *Let $x_0 \in \overline{\mathbb{F}}_q$ be such that $\mathbb{F}_q(x_0) = \mathbb{F}_{q^k}$ and $x_1, \ldots, x_{2k-1} \in \mathbb{F}_q$ be such that for all i, $x_i \neq x_j$ for all $j \neq i$. Then we define the map ψ as:*

$$\begin{aligned} \psi : \quad \mathbb{F}_q^{2k-1} &\to \mathbb{F}_{q^k} \\ (b_1, \ldots, b_{2k-1}) &\to (g(x_0)) \end{aligned}$$

where g is the unique polynomial of degree at most $2k-2$ such that for all $i \in \{1, \ldots, 2k-1\}$, $g(x_i) = b_i$.

Lemma 3.7.5. *The map ϕ defined as in definition 3.7.3 is injective.*

Proof. Suppose by contradiction that we have $z = \phi(a) = \phi(b) = y$ with $a, b \in \mathbb{F}_{q^k}$ and $a \neq b$. Then there are two polynomials g and h of degree at most $k-1$ such that $g(x_0) = a \neq b = h(x_0)$ and for all $i \in \{1, \ldots, 2k-1\}$, $g(x_i) = h(x_i)$. But this is a contradiction since there is at most one polynomial of degree at most $k-1$ passing through $2k-1$ points. □

Theorem 3.7.6. *Let $x_0 \in \overline{\mathbb{F}}_q$ be such that $\mathbb{F}_q(x_0) = \mathbb{F}_{q^k}$, $x_1, \ldots, x_{2k-1} \in \mathbb{F}_q$ be such that for all i, $x_i \neq x_j$ for all $j \neq i$ and ϕ and ψ and in definitions 3.7.3 and 3.7.4.*
Then $(2k-1, \phi, \psi)$ is a multiplication friendly embedding.

Proof. Polynomial interpolation and polynomial evaluation are $\mathbb{F}_q$-linear maps, and so are ϕ and ψ By lemma 3.7.5, ϕ is injective.
Now consider $a, b \in \mathbb{F}_{q^k}$.

$$\phi(a) = (f(x_1), \ldots, f(x_{2k-1}))$$

where $f(X)$ is a polynomial of degree at most $k-1$ such that $f(x_0) = a$.

$$\phi(b) = (g(x_1), \ldots, g(x_{2k-1}))$$

where $g(X)$ is a polynomial of degree at most $k-1$ such that $g(x_0) = b$.

$$\phi(a) * \phi(b) = (f(x_1)g(x_1), \ldots, f(x_{2k-1})g(x_{2k-1})) = (y_1, \ldots, y_{2k-1})$$

Now consider the polynomial

$$h(X) \in \mathbb{F}_q[X]_{\leq 2k-2} \text{ such that } \forall i \in \{1, \ldots, 2k-1\}, h(x_i) = y_i.$$

Since $h(X)$ has degree at most $2k-2$, it is uniquely determined by Lagrange interpolation theorem. Moreover consider $fg(X) = f(X)g(X)$. Since $f(X)$

and $g(X)$ have degree at most $k-1$, $fg(X)$ has degree at most $2k-2$ and clearly $fg(x_i) = y_i, \forall i \in \{1, \ldots, 2k-1\}$. Hence by the uniqueness of the interpolation polynomial,

$$h(X) = fg(X).$$

Thus

$$\psi(\phi(a) * \phi(b)) = \psi(h(x_1), \ldots, h(x_{2k-1})) = h(x_0) = fg(x_0) = f(x_0) \cdot g(x_0) = a \cdot b$$

□

Linear algebra properties of $\mathbb{F}_q$-linear maps

Suppose that A and B are respectively a k-dimensional and a m-dimensional $\mathbb{F}_q$ vector spaces and that ζ is a $\mathbb{F}_q$ linear map from A to B. By definition $\mathrm{Im}(\zeta)$ is a linear subspace of B. Thus, to compute $\zeta(x)$ for $x \in A$ we can proceed as follow:

- Using the $\mathbb{F}_q$ basis $(a_1, \ldots, a_k)$ of A we compute the $\mathbb{F}_q$ basis of $\mathrm{Im}(\zeta)$ $(\zeta(a_1), \ldots, \zeta(a_k))$.
- Using the $\mathbb{F}_q$ basis $(a_1, \ldots, a_k)$ of A we write x in the coordinates $(x_1, \ldots, x_k)$.
- We define the $m \times k$ matrix

$$M_\zeta = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \zeta(a_i) & \zeta(a_2) & \cdots & \zeta(a_k) \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

- We compute $\zeta(z) = M_\zeta x$.

This way of computing ζ is useful in our context because it allows us to compute a commitment to the image of an elements whose coordinates are hidden by an homomorphic commitment scheme, without opening such commitment. In fact suppose that $x = (x_1, \ldots, x_k), r = (r_1, \ldots, r_k) \in A$ and that

$$c = (\mathrm{Com}(r_1, x_1), \ldots, \mathrm{Com}(x_k, r_k)).$$

Then we can define:

$$M_\zeta \bullet c = \begin{pmatrix} \zeta(a_1)_1 & \cdots & \zeta(a_k)_1 \\ \vdots & \ddots & \vdots \\ \zeta(a_1)_m & \cdots & \zeta(a_k)_m \end{pmatrix} \bullet \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix} = \begin{pmatrix} \prod_{i=1}^{k} c_i^{\zeta(a_i)_1} \\ \vdots \\ \prod_{i=1}^{k} c_i^{\zeta(a_i)_m} \end{pmatrix} = \begin{pmatrix} d_1 \\ \vdots \\ d_m \end{pmatrix}$$

and we will have that for all $i \in \{1, \ldots, m\}$, $d_i = \mathrm{Com}(\zeta(x)_i, \zeta(r)_i)$.

Commitments over $\mathbb{F}_{q^k}$

We will now describe how a prover P can commit to an element of $\mathbb{F}_{q^k}$. Since the family of functions of such commitment scheme would be rather complicated to describe, we chose to describe it as a protocol. We assume that a homomorphic commitment scheme over $\mathbb{F}_q$ is given and that such commitment is unconditionally hiding and computationally binding. We will denote with $\mathcal{CS}$ such commitment scheme.

Protocol 3.7.7. *Given an homomorphic commitment scheme $\mathcal{CS}$ over $\mathbb{F}_q$ a prover P can commit to an element $a \in \mathbb{F}_{q^k}$ following these steps:*

Set-up phase

1. *V runs the generator of $\mathcal{CS}$*
2. *V chooses $x_0 \in \mathbb{F}_{q^k}$ such that $\mathbb{F}_q(x_0) = \mathbb{F}_{q^k}$*
3. *V chooses $x_1, \ldots, x_{2k-1} \in \mathbb{F}_q$ such that for all $i \in \{1, \ldots, 2k-1\}$, $x_i \neq x_j$ for all $j \neq i$*
4. *V sends $(pk, x_0, x_1, \ldots, x_{2k-1})$ to P*
5. *P checks that $pk \in \mathcal{PK}$ and that $x_0, \ldots, x_{2k-1}$ are correctly generated.*

Commit phase *To commit to a value $a \in \mathbb{F}_{q^k}$*

1. *P computes the coordinate-vector $(a_1, \ldots, a_k)$ using the $\mathbb{F}_q$ basis $(1, \ldots, x_0^{k-1})$ of $\mathbb{F}_{q^k}$*
2. *P selects $r \in_R \mathbb{F}_{q^k}$*
3. *P computes the coordinate-vector $(r_1, \ldots, r_k)$ using the $\mathbb{F}_q$ basis $(1, \ldots, x_0^{k-1})$ of $\mathbb{F}_{q^k}$*
4. *P computes $C = (\mathtt{commit}_{pk}(r_1, a_1), \ldots, \mathtt{commit}_{pk}(r_k, a_k))$. Such C is defined to be the commitment of a.*

Opening phase *To open a commitment P sends a and r to V*

Verify phase *To verify a opening (a, r) given a commitment $C \in \mathbb{F}_q^k$*

1. *V computes the coordinate-vector $(a_1, \ldots, a_k)$ using the $\mathbb{F}_q$ basis $(1, \ldots, x_0^{k-1})$ of $\mathbb{F}_{q^k}$*
2. *V computes the coordinate-vector $(r_1, \ldots, r_k)$ using the $\mathbb{F}_q$ basis $(1, \ldots, x_0^{k-1})$ of $\mathbb{F}_{q^k}$*
3. *V checks that for all $i \in \{1, \ldots, k\}$,$C_i \stackrel{?}{=} (\texttt{commit}_{pk}(r_i, a_i))$.*

We like to recall that, if a multiplicative friendly embedding $(2k-1, \xi, \delta)$ is given, P and V can compute $M_\xi \bullet C$ to have the commitment vector of the embedding of a in $\mathbb{F}_q^{2k-1}$. We will refer to such commitment vector as the extended commitment of a and we will denote it with $\widetilde{C}$.

Multiplication protocol over $\mathbb{F}_{q^k}$

Aim

We will now describe the protocol that proves that, given two commitments B and C of values $b, c \in \mathbb{F}_{q^k}$, a third commitment D hides the value $b \cdot c \in \mathbb{F}_{q^k}$. Given a multiplicative friendly embedding $(2k-1, \xi, \delta)$ it works as follows. P and V compute from B and C the extended commitment $\widetilde{B}$ and $\widetilde{C}$ of b and c using M_ξ. Then a component-wise multiplication proof is done using protocol 3.2.1 and the auxiliary $2k-1$ vector D' of commitments of the values $(\xi(b) * \xi(c))_i$. Finally V checks in a non interactive step that the values hidden by $M_\delta \bullet D'$ are the same as the ones hidden by D.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$, $x_0, \ldots, x_{2k-1}$ as generated in the set-up phase of the protocol 3.7.7 a friendly embedding $(2k-1, \xi, \delta)$ and commitments $B, C, D \in G \times \ldots \times G$ of b, c and bc, while the witness will be $b, c \in \mathbb{F}_{q^k}$ and the random elements $r, s, t \in \mathbb{F}_{q^k}$ that are used in protocol 3.7.7 to hide those values and their product. More precisely we say that $(x, w) \in R$ if

$$x = (pk, x_0, \ldots, x_{2k-1}, (2k-1, \xi, \delta), B, C, D) \text{ and } w = (b, r, c, s, t)$$

such that

$B = (B_1, \ldots, B_k)$ is the commitment of b according to protocol 3.7.7, with random choice r

$C = (C_1, \ldots, C_k)$ is the commitment of c according to protocol 3.7.7, with random choice s

$D = (D_1, \ldots, D_k)$ is the commitment of $b \cdot c$ according to protocol 3.7.7, with random choice t

Protocol

Protocol 3.7.8 (Multiplication protocol over $\mathbb{F}_{q^k}$)**.**
Consider the relation R as defined before, a pair of PITM (P, V) and the following protocol

Public input: $(pk, x_0, \ldots, x_{2k-1}, (2k-1, \xi, \delta), B, C, D)$

Private input: (b, r, c, s, t)

Interaction:

1. *P and V compute $\widetilde{B} = M_\xi \bullet B$ and $\widetilde{C} = M_\xi \bullet C$*
2. *P computes $r' = \xi(r)$, $t' = \xi(t)$, $s' = \xi(s)$, $b' = \xi(b)$ and $c' = \xi(c)$*
3. *For all $i \in \{1, \ldots, 2k-1\}$, P computes $D'_i = \mathrm{Com}(r'_i s'_i, b'_i c'_i)$*
4. *For all $i \in \{1, \ldots, k\}$, P computes $\rho_i = (rs)_i - t_i$*
5. *P sends D' and ρ to V*
6. *For all $i \in \{1, \ldots, 2k-1\}$, (P, V) run the protocol 3.2.1 on public input $(pk, \widetilde{B}_i, \widetilde{C}_i, D'_i)$ and private input $(b'_i, r'_i, c'_i, s'_i, r'_i s'_i)$*
7. *V checks that $M_\delta \bullet D' \stackrel{?}{=} D * (\mathrm{Com}(\rho_1, 0), \ldots, \mathrm{Com}(\rho_k, 0))$*

Theorem 3.7.9. *The pair (P, V) as described in protocol 3.7.8 is a Σ-protocol for relation R.*

Proof. Completeness: Suppose that $((pk, x_0, \ldots, x_{2k-1}, B, C, D), (b, r, c, s, t)) \in R$. Then by construction for all $i \in \{1, \ldots, 2k-1\}$, $D'_i = \mathrm{Com}(t'_i, b'_i c'_i)$, $\widetilde{B}_i = \mathrm{Com}(r'_i, b'_i)$ and $\widetilde{B}_i = \mathrm{Com}(r'_i, b'_i)$ hence the protocol 3.2.1 on step 5 will always be accepted.
Now consider D' and step 6. We have:

$$M_\delta \bullet D' = (\mathrm{Com}((\delta(r's'))_1, (\delta(b'c'))_1), \ldots, \mathrm{Com}((\delta(r's'))_k, (\delta(b'c'))_k)) =$$

$$= (\mathrm{Com}((rs)_1,(bc)_1),\ldots,\mathrm{Com}((rs)_k,(bc)_k) =$$
$$= (\mathrm{Com}(t_1,bc_1),\ldots,\mathrm{Com}(t_k,bc_k))*(\mathrm{Com}((rs)_1-t_1,0),\ldots,\mathrm{Com}((rs)_k-t_k,0)) =$$
$$= D*(\mathrm{Com}(\rho_1,0),\ldots,\mathrm{Com}(\rho_k,0))$$

Hence V will accept the protocol with probability 1.

Special Soundness: Suppose that $(pk,,x_0,\ldots,x_n,B,C,D)\in RX$, then suppose to have a collision. From it V can compute in step 6: $b',r',c',s',r's'$ and applying ξ^{-1} (ξ is injective and thus invertible in its image) b,r,c,s,rs Moreover, from ρ, V can compute $t=\rho+rs$, thus a witness.
Now suppose that $(pk,,x_0,\ldots,x_n,B,C,D)\notin RX$, then if a collision exists we can compute a witness as before, but such witness does not exist, hence a collision cannot exist either.

SHVZK: Supposing that the verifier is honest and that we have access to the simulator of the protocol 3.2.1. Then we can build a simulator $M((pk,x_0,\ldots,x_{2k-1},B,C,D))$ performing:

- M computes $\widetilde{B}=M_\xi\bullet B$ and $\widetilde{C}=M_\xi\bullet C$
- M selects for all $i\in\{1,\ldots,k\}$, $\rho'_i\in_R\mathbb{F}_q$.
- M selects E such that $M_\delta\bullet E=D*(\mathrm{Com}(\rho'_1,0),\ldots,\mathrm{Com}(\rho'_k,0))$
- M runs for all $i\in\{1,\ldots,2k-1\}$, the simulator of protocol 3.2.1 on input $(pk,\widetilde{B}_i,\widetilde{C}_i,E_i)$

The simulation in step 4 has already been proved. We just have to show that ρ' and E have the same distribution of ρ and D'. By the unconditionally hiding property of the commitment scheme, E and D' have the uniform distribution over $G\times\cdots\times G$. Clearly ρ' as the uniform distribution over $\mathbb{F}_q^k$ since it is chosen at random and independently from any other values. On the other hand, ρ depends on t, r and s, but since their distributions are uniform over $\mathbb{F}_q^k$ and independent from any other values, also ρ has the uniform distribution.

□

Proposition 3.7.10. *The protocol 3.7.8 has a communication cost of* $O(k)$

Proof. In the first 4 steps no communication are required. In step 5 P has to communicate $2k-1$ committed values and k values in $\mathbb{F}_q$, while in step 6 (P,V) uses $2k-1$ times a protocol of cost $O(1)$. Step 7 does not require communication. Thus the total communication cost is $O(k)$. □

It should be clear that from this multiplication protocol, we can adapt any other protocols described in this section to work over $\mathbb{F}_{q^k}$. This will always lead to a growth in the communication cost of a factor k.

Chapter 4

Permutation Protocols

Introduction

At this point we have introduced all the mathematical elements that we need and we are ready to restate the initial problem in mathematical terms.
We have the following situation:

- A homomorphic commitment scheme (we will use Pedersen commitment).
- Two vectors

$$x = (\texttt{commit}_{pk}(r_1, a_1), \ldots, \texttt{commit}_{pk}(r_n, a_n))$$

$$y = (\texttt{commit}_{pk}(r'_1, b_1), \ldots, \texttt{commit}_{pk}(r'_n, b_n))$$

  of committed elements $a_i, b_i \in \mathbb{F}_q$.

We want a protocol that allows P to prove to V:

- The multisets (a set with repetitions) $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ are equal.
- P knows a permutation $\tau : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that

$$\tau(i) = j \text{ if and only if } a_i = b_j.$$

Under the conditions:

- The commitment cannot be opened.
- τ cannot be revealed.

In this chapter we will provide some solutions:

Cut and Choose based protocol: Applying the method described in section 2.4.2 to our problem, we will discuss a general solution. This is in some sense our adversary: we want to build up protocols with smaller communication cost or at least with smaller cheating probability.

Permutation Matrix based Protocol: Using the fact that every permutation can be seen as a permutation matrix we describe this solution, that is already better than the Cut and Choose based.

DFT based protocol for rotation: We describe the content of the article published in 2010 by de Hoogh, Schoenmakers, Škorić and Villegas ([6]). They were able, using the Discrete Fourier Transform, to describe the problem of rotating a vector in terms of component-wise multiplication over $\mathbb{F}_q^n$ and then using known protocols to solve it.

Wedderburn decomposition based protocols: We generalize the idea of de Hoogh, Schoenmakers, Škorić and Villegas observing that the Discrete Fourier Transform can be described as an isomorphism between the group algebra of a cyclic group and $\mathbb{F}_q^n$ and that we can use any isomorphism of this type (with different groups) to obtain a similar and more general results.

4.1 Permutations, Group actions and Permutation Matrices

Before dealing with the protocols, we state some theorems that connect permutations, group actions and permutation matrices. This connections will be used in the rest of the chapter to build different types of protocols.

Definition 4.1.1 (Permutation)**.** *A permutation is a bijective map from* $\{1,\dots,n\}$ *to* $\{1,\dots,n\}$ *with* $n \in \mathbb{N}$

Theorem 4.1.2. *The set of permutations* $\{\sigma \text{ s.t. } \sigma : \{1,\dots,n\} \to \{1,\dots,n\}\}$ *is a group under the usual maps composition. Such group is denoted by* S_n

Proof. The composition of bijective maps is associative, the composition of any permutation σ with the identity permutation is σ itself and, since the permutation are bijective there exists the inverse σ^{-1} for all $\sigma \in S_n$. Thus S_n is a group. □

Permutations and Group Actions

First we define the concept of group action.

Definition 4.1.3 (Group Action)**.** *Given a group G and a set X, a group action is a function:*

$$G \times X \to X$$
$$(g, x) \to g \cdot x$$

such that it verifies:

- $1_G \cdot x = x$ *for all* $x \in X$
- $g \cdot (h \cdot x) = (gh) \cdot x$ *for all* $g, h \in G$ *and for all* $x \in X$

This definition can be interpreted in the following way, for every element g, there is a map from X to X. As proved in the following lemma, such map is bijective and hence it can be seen as a permutation on X.

Lemma 4.1.4. *Given a group G and a group action of G over the finite set X, then for all $g \in G$ the map*

$$\begin{aligned} \psi_g : \ X &\to X \\ x &\to g \cdot x \end{aligned}$$

is bijective.

Proof. Indeed suppose that ψ_g is not injective, i.e. there exists x, y such that $x \neq y$ and $g \cdot x = g \cdot y$. Then by associativity of the group action $x = (g^{-1}g) \cdot x = g^{-1} \cdot (g \cdot x) = g^{-1} \cdot (g \cdot y) = g^{-1}g \cdot y = y$. Contradiction. So ψ_g is injective and hence bijective since X is finite. □

This fact allows us to see any group G for which a group action on $\{1, \ldots, n\}$ is defined, as a subset of S_n. However to see such group as a subgroup of S_n of the same cardinality of G we need an additional property of the group action:

Definition 4.1.5 (Transitive Group Action)**.** *A group action is transitive if for all $x, y \in X$ there exists $g \in G$ such that $g \cdot x = y$*

Now the following theorem holds:

Theorem 4.1.6. *Consider a group G of cardinality n that acts transitively on $\{1, \dots, n\}$. Then G is isomorphic (as group) to a subgroup of S_n. Such isomorphism is induced by the group action.*

Proof. Consider the map φ defined as:

$$\begin{aligned} \varphi : \ G &\to S_n \\ g &\to \tau \end{aligned}$$

where τ is defined by $\tau(i) = j$ if and only if $g \cdot i = j$.
Such map is well defined since the map

$$\begin{aligned} \psi_g : \ X &\to X \\ x &\to g \cdot x \end{aligned}$$

in bijective. Moreover the map φ is injective. Indeed suppose that $\varphi(g) = \tau = \sigma = \varphi(h)$. Then for all $x \in X$,

$$g \cdot x = h \cdot x \Rightarrow h^{-1}g \cdot x = x$$

Since the action is transitive by the Burnside lemma the only element with fixed points is the identity. Hence

$$h^{-1}g = 1_G \Rightarrow g = h$$

Finally by the properties of the group action, if $\tau = \varphi(g), \sigma = \varphi(h)$ then $\varphi(gh) = \tau \circ \sigma$, $\varphi(1_G) = 1_{S_n}$ and if $\varphi(g) = \tau$ then $\varphi(g^{-1}) = \tau^{-1}$, thus the image of φ is a subgroup of S_n □

Clearly one can argue that such a transitive action may not exist, however the next property guarantees us that we can always define a transitive action on $\{1, \dots, n\}$ for a group of cardinality n, just using the properties of the operation of the group.

Proposition 4.1.7. *Given any group G of cardinality n, then a transitive group action of G on $\{1, \dots, n\}$ can be defined.*

Proof. List of the elements of $G = \{g_1, \dots, g_n\}$. Then define:

$$\begin{aligned} G \times \{1, \dots, n\} &\to \{1, \dots, n\} \\ (g, x) &\to y \end{aligned}$$

where y is such that $gg_x = g_y$. Clearly $1_G \cdot x = x$ for all $x \in \{1, \dots, n\}$ and $g \cdot (h \cdot x) = gh \cdot x$ for all $g, h \in G$ and $x \in \{1, \dots, n\}$. Moreover for all $a, b \in \{1, \dots, n\}$ we have $g \cdot a = b$ for $g = g_b g_a^{-1}$ and hence such action is transitive. □

Hence, combining theorem 4.1.6 and proposition 4.1.7 we have that:

Corollary 4.1.8. *Any group G of cardinality n is (isomorphic to) a subgroup of S_n*

Permutations and Permutation Matrices

Now we relate permutations with matrices. Several equivalent definitions of a permutation matrix can be done. Not surprisingly, we will have a one to one correspondence between permutation matrices and permutations.

Definition 4.1.9 (Permutation Matrix)**.** *A matrix $M \in \mathcal{M}(n, \mathbb{F}_2)$ is called a permutation matrix if it can be obtained from the identity matrix by permuting its rows according to a permutation $\sigma \in S_n$. Equivalently M is a permutation if it has exactly one non-zero component in every row and in every column and such components are all equal to 1.*

Theorem 4.1.10. *The set $\{P \in \mathcal{M}(n, \mathbb{F}_2) : P \text{ is a permutation matrix}\}$ is a group under usual matrix multiplication. Such group is denoted by $\mathcal{P}(n)$*

Proof. Clearly the identity matrix is a permutation matrix and the matrices multiplication is associative. Moreover one can checks directly that the product of two permutation matrices is still a permutation matrix and that if a permutation matrix is obtained permuting the rows of the identity according to σ, the inverse can be obtained by permuting the rows of the identity according to σ^{-1}. □

Theorem 4.1.11. *The group $\mathcal{P}(n)$ is isomorphic to S_n*

Proof. The proof is trivial since every permutation matrix is uniquely defined by a permutation and viceversa. Moreover the matrices multiplication of $\mathcal{P}(n)$ corresponds to the composition of permutations of S_n. □

4.2 Cut and Choose-based Protocol

Aim

The aim of this protocol is to allow P to prove that, given two n-vectors x and y of committed values a and b, he knows a permutation τ, that applied to the indices of the components sends a to b.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$ and two n-vectors of commitments $x, y \in G \times \ldots \times G$ while the witness will be the two vectors of values $a, b \in \mathbb{F}_q^n$, the n-vectors of random elements $r, r' \in \mathbb{F}_q^n$ that hide those values and the permutation $\tau \in S_n$. More precisely we say that $(x, w) \in R$ if

$$x = (pk, x, y) \text{ and } w = (a, r, b, r', \tau)$$

such that for all $i \in \{1, \ldots, n\}$

$$x_i = \texttt{commit}_{pk}(r_i, a_i), \quad y_i = \texttt{commit}_{pk}(r'_i, b_i), \quad \tau(i) = j \text{ if and only if } a_i = b_j.$$

Protocol

Protocol 4.2.1. *Consider the relation R as before, a pair of PITMs (P, V) and the following protocol:*

Public input (pk, x, y)

Private input (a, r, b, r', τ)

Interaction

1. *P selects $\sigma \in_R S_n$*
   P selects for all $i, s_i \in_R \mathbb{F}_p$
   P computes $z_i = \mathrm{Com}(a_{\sigma(i)}, s_i)$
   P sends $(z_1, \ldots, z_n)$ to V.
2. *V selects $e \in_R \{0, 1\}$*
   V sends e to P

3. *if* $e = 0$,
   P *computes for all* i, $t_i = s_i - r_{\sigma(i)}$
   P *sends* σ *and* $t_1, \ldots, t_n$
   if $e = 1$,
   P *computes for all* i, $t'_i = s_i - r_{\sigma\tau^{-1}(i)}$
   P *sends* $\sigma\tau^{-1}$ *and* $t'_1, \ldots, t'_n$
4. *For all* i, V *checks either*

$$z_i \stackrel{?}{=} x_{\sigma(i)} \cdot \mathrm{Com}(t_1, 0)$$

   or

$$z_i \stackrel{?}{=} x_{\sigma\tau^{-1}(i)} \cdot \mathrm{Com}(t'_1, 0)$$

Theorem 4.2.2. *The pair* (P, V) *as described in protocol 4.2.1 is a* Σ*-protocol for relation* R*.*

Proof. This comes directly from theorem 2.4.3. □

Proposition 4.2.3. *The protocol 4.2.1 has a communication cost of* $O(n)$ *with a cheating probability of* $\frac{1}{2}$ *or* $O(n^2)$ *if we want a cheating probability negligible in* n*.*

Proof. (P,V) has to communicate

$$(z_1, \ldots, z_n, e, h, t_1, \ldots, t_n)$$

or

$$(z_1, \ldots, z_n, e, g^{-1}h, t'_1, \ldots, t'_n).$$

Thus the communication cost is $O(n)$. If we want a cheating probability negligible in n we have to repeat the protocol n times, thus the final cost would be $O(n^2)$. □

4.3 Permutation Matrix-based Protocol

Aim

In this section we will use the presentation of a permutation as a permutation matrix to prove that two given two n-vectors of commitments x and y, the n-vectors a and b hidden by them are one the permutation of the other.

This protocol will be composed of two main steps: the verification of the presentation (i.e. that the committed $n \times n$ matrix M hides a permutation matrix P) and the verification of the multiplication (i.e Mx hides the same vector hidden by y). These steps will be performed with the protocols 3.6.1 and 3.3.1

Relation

Our public input will be a public key $pk \in \mathcal{PK}$, two n-vectors of commitment $x, y \in G \times \ldots \times G$ and a matrix of commitments $M \in \mathcal{M}(n, G)$ while the witness will be the two vectors of values $a, b \in \mathbb{F}_q^n$, the n-vectors of random elements $r, r' \in \mathbb{F}_q^n$ that hide those values, the permutation matrix $Q \in \mathcal{M}(n, \mathbb{F}_q)$ hidden by M and the corresponding matrix S of random values that hides it. More precisely we say that $(x, w) \in R$ if

$$x = (pk, x, y, M) \text{ and } w = (a, r, b, r', Q, S)$$

such that for all $i \in \{1, \ldots, n\}$

$$x_i = \texttt{commit}_{pk}(r_i, a_i), \quad y_i = \texttt{commit}_{pk}(r'_i, b_i)$$

for all $i, j \in \{1, \ldots, n\}$

$$M_{ij} = \texttt{commit}_{pk}(S_{ij}, Q_{ij})$$

and

$$Qa = b$$

Protocol

Protocol 4.3.1 (Permutation Matrix based Protocol)**.** *Consider the relation R as defined before, a pair of PITM (P, V) and the following protocols run in parallel with the same challenge $e \in \mathbb{F}_q$:*

Public Input: (pk, x, y, M)

Private Input: (a, r, b, r', Q, S)

Interaction:

1. (a) *For all* $i \in \{1, \ldots, n\}$, (P, V) *runs the protocol 3.6.1 on public input* (pk, M_i) *and private input* (k_i, S_i), *where* M_i *and* S_i *are the* i^{th} *rows of* M *and* S *and* k_i *is the column index of the non-zero component of the* i^{th} *row of* Q.
   (b) *For all* $j \in \{1, \ldots, n\}$, (P, V) *runs the protocol 3.6.1 on public input* (pk, M_j) *and private input* (k_j, S_j), *where* M_j *and* S_j *are the* j^{th} *columns of* M *and* S *and* k_j *is the row index of the non-zero component of the* j^{th} *column of* Q.
2. *For all* $i \in \{1, \ldots, n\}$, (P, V) *runs the protocol 3.3.1 on public input* (pk, M_i, x, y_i) *and private input* (Q_i, S_i, x, r, r'_i), *where* M_i, Q_i *and* S_i *are the* i^{th} *rows of* M, Q *and* S.

Theorem 4.3.2. *The pair* (P, V) *as described in protocol 4.3.1 is a* Σ*-protocol for relation* R.

Proof. The theorem comes from the fact that we are simply running in parallel subprotocols that we proved to be Σ-protocols □

Proposition 4.3.3. *The protocol 4.3.1 has a communication cost of* $O(n^2)$

Proof. (P, V) is running $2n$ times the protocol 3.6.1 of cost $O(n)$ and n times the protocol 3.3.1 of cost $O(n)$, hence the total communication cost is $O(n^2)$. □

4.4 DFT-based Protocol

Discrete Fourier Transform

Definition 4.4.1 (Discrete Fourier Transform)**.** *Suppose* q *is prime and suppose that* $n|q-1$, *then we define as Discrete Fourier transform over the finite field* $\mathbb{F}_q$, *and we call it* D_n, *the function:*

$$\begin{aligned} D_n : \ \mathbb{F}_q[X]_{\leq n-1} &\to \bigoplus_{i=1}^{n} \mathbb{F}_q \\ f &\to (f(\beta^0), f(\beta^1), \ldots, f(\beta^{n-1})) \end{aligned}$$

where β *is such that* $\beta^n = 1$ *and* $\beta^k \neq 1$ *if* $k < n$.

Proposition 4.4.2. D_n *is an isomorphism of* $\mathbb{F}_q$*-vector spaces.*

Proof. It comes directly from theorem 3.7.2 setting for all $i \in \{1, \dots, n\}$, $x_i = \beta^{i-1}$. □

Observation 4.4.3. Given an element $(x_0, \dots, x_{n-1}) \in \mathbb{F}_q^n$, we can see it as the coefficient-vector of a polynomial in $\mathbb{F}_q[X]_{\leq n-1}$. Hence we can apply the Discrete Fourier transform to it and we get:

$$\begin{array}{rccc} D_n : & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & (x_0, \dots, x_{n-1}) & \to & (x'_0, \dots, x'_{n-1}) \end{array}$$

where

$$x'_i = \sum_{j=0}^{n-1} x_j \beta^{ij}.$$

Definition 4.4.4. *Using observation 4.4.3 we can define D_n as the matrix:*

$$\begin{pmatrix} \beta^0 & \beta^0 & \beta^0 & \dots & \beta^0 \\ \beta^0 & \beta^1 & \beta^2 & \dots & \beta^{(n-1)} \\ \beta^0 & \beta^2 & \beta^4 & \dots & \beta^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta^0 & \beta^{(n-2)} & \beta^{2(n-2)} & \dots & \beta^{(n-1)(n-2)} \\ \beta^0 & \beta^{(n-1)} & \beta^{2(n-1)} & \dots & \beta^{(n-1)^2} \end{pmatrix}$$

or more synthetically $(b_{ij}) = (\beta^{(i-1)(j-1)})$

Proposition 4.4.5. *The matrix that defines D_n has inverse:*

$$\frac{1}{n} \begin{pmatrix} \beta^0 & \beta^0 & \beta^0 & \dots & \beta^0 \\ \beta^0 & \beta^{-1} & \beta^{-2} & \dots & \beta^{-(n-1)} \\ \beta^0 & \beta^{-2} & \beta^{-4} & \dots & \beta^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta^0 & \beta^{-(n-2)} & \beta^{-2(n-2)} & \dots & \beta^{-(n-1)(n-2)} \\ \beta^0 & \beta^{-(n-1)} & \beta^{-2(n-1)} & \dots & \beta^{-(n-1)^2} \end{pmatrix}$$

or more synthetically $(c_{ij}) = \frac{1}{n}(\beta^{-(i-1)(j-1)})$

Proof. Consider the matrix product (d_{ij}).

$$d_{ij} = \frac{1}{n} \sum_{k=1}^{n} b_{ik} \cdot c_{kj} = \frac{1}{n} \sum_{k=0}^{n-1} \beta^{k(i-j)}$$

hence we have

$$d_{ij} = \begin{cases} \frac{1}{n}\sum_{k=0}^{n-1}\beta^0 = 1 & \text{if } i = j \\ \\ \frac{1}{n}\sum_{k=0}^{n-1}\beta^{k(i-j)} = 0 & \text{if } i \neq j \end{cases}$$

Where the last holds because $\beta^n = 1$, $\beta^{(i-j)} \neq 1$ for all $i, j \in \{1, \ldots, n\}$, $i \neq j$ and

$$\frac{1}{n}\sum_{k=0}^{n-1}\beta^{kt} = \frac{1-(\beta^{(i-j)})^n}{n(1-\beta^{(i-j)})} = \frac{1-1^{(i-j)}}{n(1-\beta^{(i-j)})} = 0$$

□

Rotation and DFT

Now we restrict our initial problem to the case of rotations and we see how use the properties of the DFT to construct a protocol that solves it, using the protocols of chapter 3.

Definition 4.4.6 (Rotation)**.** *A rotation σ is a permutation that for all $i \in \{1, \ldots, n\}$ satisfies:*

$$\sigma(i) = i + r \mod n$$

where $r \in \mathbb{Z}_n$.

Proposition 4.4.7. *Consider two vectors $(x_0, \ldots, x_{n-1})$ and $(y_0, \ldots, y_{n-1})$ such that for all $i \in \{0, \ldots, n-1\}$, $x_i = y_{\sigma(i)}$ for a rotation σ. Then we have*

$$y'_k = \alpha^k x'_k \quad \forall k \in \{1, \ldots, n\}$$

where $\alpha = \beta^r$ and r is such that $\sigma(i) = j + r \bmod n$.

Proof. We have

$$y'_k = \sum_{j=0}^{n-1} y_j \beta^{kj} = \sum_{j=0}^{n-1} x_{j-r}\beta^{kj} = \sum_{i=0}^{n-1} x_i \beta^{k(i+r)} = \alpha^k x'_k$$

□

Now suppose that we have two vectors

$$x = (\texttt{commit}_{pk}(r_{a_0}, a_0), \ldots, \texttt{commit}_{pk}(r_{a_{n-1}}, a_{n-1}))$$

$$y = (\texttt{commit}_{pk}(r_{b_0}, b_0), \ldots, \texttt{commit}_{pk}(r_{b_{n-1}}, b_{n-1}))$$

of committed elements $a_i, b_i \in \mathbb{F}_q$ such that for all $i \in \{0, \ldots, n-1\}$ $a_i = b_{\sigma(i)}$ for a rotation σ.
Since β is public and the DFT is $\mathbb{F}_q$-linear, a verifier can compute:

$$x' = (\prod_{j=0}^{n-1} x_j^{\beta^{0j}}, \ldots, \prod_{j=0}^{n-1} x_j^{\beta^{(n-1)j}}) =$$

$$= (\mathrm{Com}(\sum_{j=0}^{n-1} r_{a_j}\beta^{0j}, \sum_{j=0}^{n-1} a_j\beta^{0j}), \ldots, \mathrm{Com}(\sum_{j=0}^{n-1} r_{a_j}\beta^{(n-1)j}, \sum_{j=0}^{n-1} a_j\beta^{(n-1)j})) =$$

$$= (\mathrm{Com}(r'_{a_0}, a'_0), \ldots, \mathrm{Com}(r'_{a_{n-1}}, a'_{n-1}))$$

and

$$y' = (\prod_{j=0}^{n-1} y_j^{\beta^{0j}}, \ldots, \prod_{j=0}^{n-1} y_j^{\beta^{(n-1)j}}) =$$

$$= (\mathrm{Com}(\sum_{j=0}^{n-1} r_{b_j}\beta^{0j}, \sum_{j=0}^{n-1} b_j\beta^{0j}), \ldots, \mathrm{Com}(\sum_{j=0}^{n-1} r_{b_j}\beta^{(n-1)j}, \sum_{j=0}^{n-1} b_j\beta^{(n-1)j})) =$$

$$= (\mathrm{Com}(r'_{b_0}, b'_0), \ldots, \mathrm{Com}(r'_{b_{n-1}}, b'_{n-1}))$$

Now if a prover can prove to the verifier that for all $i \in \{0, \ldots, n-1\}$:

$$x'_i = (y'_i)^{\alpha^i} \tag{4.1}$$

keeping α secret, then he would also prove to the verifier that

$$a'_i = (b'_i\alpha^i)$$

and by proposition 4.4.7 that

$$a_i = b_{i+r}$$

without revealing r (and hence the rotation σ).

The main idea

The Σ-protocol to prove equation (4.1) is described in the article [6] and it can be done using the multiplication protocol. However we are much more interested in the idea behind this construction.

Idea 4.4.8. *To prove a relation between x and y de Hoogh, Schoenmakers, Škorić and Villegas find a way to translate the problem into a domain where such relation can be proved component-wise and protocols to do it were already known. The fact that they use the DFT for that is not so important. Every isomorphismfrom the usual domain to a domain in which we can use known protocols can be applied in the same way.*

In the following section we will introduce some notions from representation theory that allow us to see the DFT from a different point of view and then to generalize it easily.

4.5 A new point of view

All the procedures described in the previous section can be interpreted in a different way, that suggests a good strategy to generalize the idea of [6], applying it not only to the rotation but to any permutation of a given group (seen as a subgroup of S_n). In this section we will state some basic results of representation theory of finite groups (the proofs can be found in [3] or any other basic book on finite representation theory), that will allow us to give a new point of view on the DFT-based protocol.

Definition 4.5.1 (Group Algebra)**.** *Let G be a finite group and K a field, we define the group algebra $K[G]$ as the vector space over K with basis G. Hence every element $x \in K[G]$ can be written as a formal sum of the following form:*

$$x = \sum_{g \in G} x_g g$$

with $x_g \in K, g \in G$

Proposition 4.5.2. *The group algebra is a ring with component-wise addiction and multiplication defined as*

$$x \cdot y = \left(\sum_{g \in G} x_g g\right) \cdot \left(\sum_{h \in G} y_h h\right) = \sum_{g \in G} \left(\sum_{h \in G} (x_g y_h)(gh)\right) = \sum_{g \in G} \left(\sum_{h \in G} x_{gh^{-1}} y_h\right) g$$

Theorem 4.5.3. *Let G be a finite group and $\mathbb{F}_q$ a finite field of characteristic q such that $|G|$ does not divide q. Then*

$$\mathbb{F}_q[G] \cong \bigoplus_{i=1}^{t} \mathcal{M}(n_i, \mathbb{F}_{q^{k_i}})$$

as algebras and

$$\sum_{i=1}^{t} n_i^2 k_i = \dim_{\mathbb{F}_q} \mathbb{F}_q[G] = |G|$$

Observation 4.5.4. Since the isomorphism of theorem 4.5.3 is an isomorphism of algebras, it maps elements of $\mathbb{F}_q[G]$ in $\bigoplus_{i=1}^{t} \mathcal{M}(n_i, \mathbb{F}_{q^{k_i}})$ in such a way that the operations are respected. Thus the product of two elements in $\mathbb{F}_q[G]$ is mapped in the component-wise product (the components are the matrices) of the image of the two elements.

The isomorphism of theorem 4.5.3 can be founded explicitly, if the group G is known. In fact it depends on the irreducible representations and on the irreducible characters of the group G. The study of how to compute this character and the corresponding isomorphism is beyond the aim of this thesis, however many books about this topic can be found in the literature and there are even computer algebra packages, such as the GAP package *Wedderga*, (more information can be founded here [14]) that can do the computations.

Notation 4.5.5. From now on we will denotes with χ this isomorphism of algebras and we will assume that is $\mathbb{F}_q$ linear. Moreover we will refer to $\bigoplus_{i=1}^{t} \mathcal{M}(n_i, \mathbb{F}_{q^{k_i}})$ as to the Wedderburn decomposition of $\mathbb{F}_q[G]$

Proposition 4.5.6. *If G is abelian and $|G|$ divides $q-1$, then*

$$\mathbb{F}_q[G] \cong \mathbb{F}_q^{|G|}$$

Proposition 4.5.7. *Suppose that G is a cyclic (hence abelian) finite group of order n and that $\mathbb{F}_q$ is a finite field of characteristic q. Suppose also that $n|q-1$. Then the isomorphism*

$$\chi : \mathbb{F}_q[G] \to \mathbb{F}_q^n$$

is the Discrete Fourier Transform.

Loosely speaking the proposition 4.5.7 holds because the irreducible characters of a cyclic group of order n are powers of a primitive n-root of unit. Thus χ and D_n map the element $(x_0, \ldots, x_{n-1})$ (seen as the coefficient vector of an element of $\mathbb{F}_q[G]$ for χ and as a coefficients vector of a polynomial of degree at most $n-1$ for D_n) in the same element $(x'_0, \ldots, x'_{n-1}) \in \mathbb{F}_q^n$.

Now it should be clear that we can look at the solution of de Hoogh, Schoenmakers, Škorić and Villegas in a different way. In fact we can interpret a and b as the coefficients vectors of two elements in the group algebra $\mathbb{F}_q[G]$, $D_n(a)$ and $D_n(b)$ as $\chi(a)$ and $\chi(b)$, $(\alpha^0, \ldots, \alpha^{n-1})$ as $\chi(\beta)$ and the proof that $a_i = \alpha^i b_i$ as the proof that $\chi(b) * \chi(\beta) = \chi(a)$. The advantage of looking at this method from the point of view of representation theory is that it is natural to ask if we can use a similar method, when G is not cyclic. In fact this would mean that we can prove in zero-knowledge, not only the rotation of a vector, but also a generic permutation, provided that it is in the established group G.

Observation 4.5.8. There are many differences that we have to face when G is not cyclic.

- At the beginning we will only consider groups G with cardinality equal to the number n of components of our starting vector. In this way the group algebra $\mathbb{F}_q[G]$ has dimension n over $\mathbb{F}_q$ and hence the identification of the starting vector with an element of $\mathbb{F}_q[G]$ is straightforward. Later we will give some conditions to use groups that have cardinality greater then n.

- The operation that we have to perform in the Wedderburn decomposition is not just the multiplication of a root of unit, but we have to multiply by something different, namely by the element that will correspond (in a way that we will explain later) in the Wedderburn decomposition to the permutation σ that we want to apply. Hence we have to use the protocols to perform the multiplication proof and the inner product proof.

- The Wedderburn decomposition contains not only copies of $\mathbb{F}_q$ but also field extensions $\mathbb{F}_{q^{k_i}}$. Hence we have to use the section 3.7 protocols to perform the multiplication proof and the inner product proof in the field extension.

- The Wedderburn decomposition contains not only extension of our field, but also matrix algebras on such extensions. Hence we have to use the matrix multiplication protocol.

4.6 Wedderburn-based Protocol

Suppose that we have two vectors $x, y \in \mathbb{F}_q^n$ and that there exists a permutation $\tau : \{1, \dots, n\} \to \{1, \dots, n\}$ such that $x_i = y_{\tau(i)}$. Moreover suppose that we have a group $G = \{g_1, \dots, g_n\}$ of cardinality n such that there exists a $g \in G$ that verifies the following:

$$gg_i = g_j \text{ if and only if } \tau(i) = j$$

Basically we are saying that if we see G as a subgroup of S_n as in section 4.1, then $\tau \in G$. Then, if we see x, y and g as elements of $\mathbb{F}_q[G]$ the following proposition holds:

Proposition 4.6.1. *Suppose that we have $x = \sum_{i=1}^n x_{g_i} g_i$ and $y = \sum_{i=1}^n y_{g_i} g_i$ with $x, y \in \mathbb{F}_q[G]$, such that $y_{g \cdot g_i} = x_{g_i} \forall i \in \{1, \dots, n\}$. Then*

$$g \cdot x = y$$

where $\cdot$ is the usual multiplication in $\mathbb{F}_q[G]$

Proof. By the multiplication in the group algebra we have:

$$g \cdot x = g \cdot \sum_{i=1}^n x_{g_i} g_i = \sum_{i=1}^n x_{g_i} g g_i = \sum_{i=1}^n x_{g^{-1} g_i} g_i = \sum_{i=1}^n y_{g_i} g_i = y$$

□

This proposition allows us to interpret the permutation $\tau \in S_n$ of a vector $x \in \mathbb{F}_q^n$ as the multiplication of the corresponding element $\bar{x} = \sum_{i=1}^n x_{g_i} g_i$ in the group algebra $\mathbb{F}_q[G]$ with the element g of the group G corresponding to τ (provided that it is in the group). However proving multiplications in the group algebra is not efficient. Thus, as for the DFT, we will consider the relation between $\bar{x}, \bar{y}$ and g in the Wedderburn decomposition of $\mathbb{F}_q[G]$. In fact, since χ is an isomorphism of algebras, the following corollary holds:

Corollary 4.6.2. *Suppose that we have* $x = \sum_{i=1}^{n} x_{g_i} g_i$ *and* $y = \sum_{i=1}^{n} y_{g_i} g_i$ *with* $x, y \in \mathbb{F}_p[G]$*, such that* $y_{g \cdot g_i} = x_{g_i} \forall i \in \{1, \ldots, n\}$*. Then*

$$\chi(g) * \chi(x) = \chi(y)$$

where $*$ *is the component-wise product in* $\bigoplus_{i=1}^{t} \mathcal{M}(n_i, \mathbb{F}_{q^{k_i}})$ *(such components are matrices).*

Before describing formally the protocol, we will set some notations:

Notation 4.6.3.

- Given an homomorphic commitment scheme, any $\mathbb{F}_q$ linear map ζ induces a map that can be applied to committed values. We will denote such induced map with $\widehat{\zeta}$ without describing it explicitly.
- Sometimes we will apply a function with domain $\mathbb{F}_q[G]$ to an element $a \in \mathbb{F}_q^n$. We assume to be applying it to $\sum_{i=1}^{n} a_i g_i$.
- We can see any element $g \in G$ as the element $\sum_{i=1}^{n} a_i g_i \in \mathbb{F}_q[G]$ where $a_i = 0$ if $g \neq g_i$ and $a_i = 1$ if $g = g_i$. We will denote by $\bar{g}$ the vector $(a_1, \ldots, a_n) \in \mathbb{F}_q^n$
- In the following protocols we will have the group G selected by the generator of the commitment scheme and the group of permutations H.

Relation

Our public input will be a public key $pk \in \mathcal{PK}$, two n-vectors of commitments $x, y \in G \times \ldots \times G$ and a group $H = \{h_1, \ldots, h_n\}$, while the witness will consist of the two vectors of values $a, b \in \mathbb{F}_q^n$, the n-vectors of random elements $r, r' \in \mathbb{F}_q^n$ that hide those values and the element $h_k \in H$ that corresponds to the permutation. More precisely we say that $(x, w) \in R$ if

$$x = (pk, x, y, H) \text{ and } w = (a, r, b, r', h_k)$$

such that for all $i \in \{1, \ldots, n\}$

$$x_i = \texttt{commit}_{pk}(r_i, a_i), \quad y_i = \texttt{commit}_{pk}(r'_i, b_i)$$

$$a_i = b_j \text{ if and only if } h_k h_i = h_j$$

Protocol

Protocol 4.6.4. *Consider the relation R as defined before, a pair of PITM (P, V) and the following protocols run in parallel with the same challenge $e \in \mathbb{F}_q$:*

Public Input: (pk, x, y, H)

Private Input: (a, r, b, r', h_k)

Interaction:

1. *P computes the vector $\bar{h_k}$ (observe that the non zero component will be k).*
   For all $i \in \{1, \ldots, n\}$, P selects $w_i \in_R \mathbb{F}_q$
   For all $i \in \{1, \ldots, n\}$, P computes $\gamma_i = \mathrm{Com}(w_i, (\bar{h_k})_i)$
   P sends γ to V
   (P, V) uses the protocols 3.6.1 *on public input (pk, γ) and private input (k, w) to checks that γ hides an element of H.*
2. *V computes $\widehat{\chi}(x), \widehat{\chi}(y)$ and $\widehat{\chi}(\gamma)$*
   For every component $i \in \{1, \ldots, t\}$ of the Wedderburn decomposition of $\mathbb{F}_q[H]$, (P, V) runs the protocol 3.4.1 on public input

   $$(pk, (\widehat{\chi}(x))_i, (\widehat{\chi}(\gamma))_i, (\widehat{\chi}(y))_i)$$

   and private input

   $$((\chi(a))_i, (\chi(r))_i, (\chi(\bar{h_k}))_i, (\chi(w))_i, (\chi(r'))_i)$$

   to check the component-wise product in the Wedderburn decomposition, where the components are matrices

Theorem 4.6.5. *The pair (P, V) as described in protocol 4.6.4 is a public-coin relaxed Σ-protocol for relation R.*

Proof. All the properties of the protocol 4.6.4 comes from protocol 3.6.1 and protocol 3.4.1 since all the other steps are non iterative, thus it is a public-coin relaxed Σ-protocol for relation R. □

Proposition 4.6.6. *The protocol 4.6.4 has a communication cost of $O(n)$*

Proof. The step 1 has communication cost of $O(n)$ since we are using once the protocol 3.6.1. In the second step we perform a number of matrix multiplication proofs equal to the number of components in the Wedderburn decomposition. Such matrices have entries in $\mathbb{F}_{q^{k_i}}$ and dimension n_i. But by theorem 4.5.3

$$\sum_{i=1}^{t} n_i^2 k_i = |G| = n$$

Hence the total cost is $O(n)$. □

4.7 Generalized Wedderburn-based Protocol

The protocol 4.6.4 has the big advantage over the protocol presented in [6] that can be used not just for proving rotations, but also general permutations. However, once that the group G has been chosen the number of possible permutations is always n, as the number of possible rotations. Depending on the application, it can be useful to be able to use larger groups of permutations. The problem is the following: in protocol 4.6.4 it is trivial to see an element $x = (x_1, \ldots, x_n)$ of $\mathbb{F}_q^n$ as an element of the group algebra $\mathbb{F}_q[G]$ just setting $\bar{x} = \sum_{i=1}^{n} x_i g_i$. This cannot be done when $|G| > n$. Thus we need a way to embed $\mathbb{F}_q^n$ in $\mathbb{F}_q[G]$. Moreover, to build protocol, we also need a map to "come back" that respects the multiplication in the group algebra. To define such embedding we will first define a proper embedding of X in G, where the terms "proper" denotes the fact that there is a map from G to X that respects the multiplication in G. Formally we have the following definition:

Definition 4.7.1 (Proper embedding)**.** *Let G be a group acting on $X = \{1, \ldots, n\}$. We call a proper embedding of X in G a pair of functions $\sigma : G \to X$ and $\rho : X \to G$ that satisfy*

$$\sigma(g \cdot \rho(i)) = g(i)$$

for all $g \in G$ and $i \in X$.

The following theorem gives us a sufficient and necessary condition to have a proper embedding of X in G.

Theorem 4.7.2. *Given a group G and the set $X = \{1, \ldots, n\}$, a proper embedding of X in G exists if and only if there exists a transitive group action of G on X.*

Proof. Suppose that a proper embedding of X in G exists. For any i and any j consider $g = \rho(j)\rho(i)^{-1}$. We have:

$$g(i) = \sigma(g \cdot \rho(i)) = \sigma(\rho(j)\rho(i)^{-1} \cdot \rho(i)) = \sigma(1_G \cdot \rho(j)) = 1_G(j) = j$$

Hence for every $i, j \in X$ there exists a $g \in G$ such that $g(i) = j$. Thus the action is transitive.
Now suppose that there exists a transitive action of G on X. For all i in X choose an element of G that maps 1 to i and call it g_i. Such elements exists by transitivity. Define the proper embedding as:

$$\sigma(g) = g(1)$$

$$\rho(i) = g_i$$

ρ is well defined by the fact that we choose only one g for each i. σ is well defined because if $g_1 = h_1$ then

$$\sigma(g) = g(1) = h(1) = \sigma(h)$$

moreover it is defined over the whole G since the only g that send 1 in 1 is the identity. Finally

$$\sigma(g \cdot \rho_i) = \sigma(gg_i) = gg_i(1) = g(i)$$

hence σ and ρ are a proper embedding. □

Now, starting from a proper embedding of X in G we construct a proper embedding of $\mathbb{F}_q^n$ in $\mathbb{F}_q[G]$ as follows:

Definition 4.7.3. *Let σ and ρ be a proper embedding of X in G, then we define a proper embedding of $\mathbb{F}_q^n$ in $\mathbb{F}_q[G]$ as the pair of functions:*

$$\begin{aligned} \sigma' : \quad \mathbb{F}_q^n \quad &\to \mathbb{F}_q[G] \\ (x_1, \ldots, x_n) \quad &\to \sum_{g \in G} x_{\sigma(g)} g \end{aligned}$$

and

$$\begin{aligned} \rho' : \quad \mathbb{F}_q[G] \quad &\to \mathbb{F}_q^n \\ (\textstyle\sum_{g \in G} x_g g) \quad &\to (x_{\rho(1)}, \ldots, x_{\rho(n)}) \end{aligned}$$

Proposition 4.7.4. *σ' and ρ' are $\mathbb{F}_q$-linear and*

$$\rho'(g \cdot \sigma'(x)) = (x_{g^{-1}(1)}, \ldots, x_{g^{-1}(n)})$$

Proof. Clearly they are both $\mathbb{F}_q$-linear by definition. Then

$$\rho'(g \cdot \sigma'(x)) = \rho'(g \cdot \sum_{h \in G} x_{\rho(h)} h) = \rho'(\sum_{h \in G} x_{g^{-1}\rho(h)} h) = (x_{g^{-1}(1)}, \ldots, x_{g^{-1}(n)}).$$

□

Relation

Our public input will be a public key $pk \in \mathcal{PK}$, two n-vectors of commitments $x, y \in G \times \ldots \times G$ and a group $H = \{h_1, \ldots, h_m\}$ of cardinality m acting transitively on $X = \{1, \ldots, n\}$, while the witness will be the two vectors of values $a, b \in \mathbb{F}_q^n$, the n-vectors of random elements $r, r' \in \mathbb{Z}_q^n$ that hide those values and the element $h_k \in H$ that corresponds to the permutation. More precisely we say that $(x, w) \in R$ if

$$x = (pk, x, y, H) \text{ and } w = (a, r, b, r', h_k)$$

such that for all $i \in \{1, \ldots, n\}$

$$x_i = \texttt{commit}_{pk}(r_i, a_i), \quad y_i = \texttt{commit}_{pk}(r'_i, b_i)$$

$$a_i = b_j \text{ if and only if } h_k \cdot i = j$$

Protocol

Protocol 4.7.5. *Consider the relation R as defined before, a pair of PITM (P, V) and the following protocols run in parallel with the same challenge $e \in \mathbb{F}_q$:*

Public Input: (pk, x, y, H)

Private Input: (a, r, b, r', h_k)

Interaction:

1. *P computes $\bar{a} = \sigma'(a)$, $\bar{b} = \sigma'(b)$, $\bar{r} = \sigma'(r)$ and $\bar{r}' = \sigma'(r')$.*

2. *P and V compute both $\bar{x} = \widehat{\sigma'}(x)$ and $\bar{y} = \widehat{\sigma'}(y)$*
3. *P computes the vector $\bar{h_k}$ (observe that the non zero component will be the k^{th}).*
   For all $i \in \{1, \ldots, m\}$, P selects $w_i \in_R \mathbb{F}_q$
   For all $i \in \{1, \ldots, m\}$, P computes $\gamma_i = \mathrm{Com}(w_i, (\bar{h_k})_i)$
   P sends γ to V
   (P, V) uses the protocol 3.6.1 *on public input (pk, γ) and private input (k, w) to check that γ hides an element of H.*
4. *V computes $\widehat{\chi}(\bar{x}), \widehat{\chi}(\bar{y})$ and $\widehat{\chi}(\gamma)$*
   For every component $i \in \{1, \ldots, t\}$ of the Wedderburn decomposition of $\mathbb{F}_q[H]$, (P, V) runs the protocol 3.4.1 (extended to works on $\mathbb{F}_q$ extensions) on public input

$$(pk, (\widehat{\chi}(\bar{x}))_i, (\widehat{\chi}(\gamma))_i, (\widehat{\chi}(\bar{y}))_i)$$

   and private input

$$((\chi(\bar{a}))_i, (\chi(\bar{r}))_i, (\chi(\bar{h_k}))_i, (\chi(w))_i, (\chi(\bar{r}'))_i)$$

   to check the component-wise product in the Wedderburn decomposition, where the components are matrices

Theorem 4.7.6. *The pair (P, V) as described in protocol 4.7.5 is a public-coin relaxed Σ-protocol for relation R.*

Proof. All the properties of the protocol 4.7.5 come from protocol 3.6.1 and protocol 3.4.1 since all the other steps are non iterative, thus it is a public-coin relaxed Σ-protocol for relation R. □

Proposition 4.7.7. *The protocol 4.7.5 has a communication cost of $O(|H|)$*

Proof. Step 1 and Step 2 do not require communication. Step 3 has communication cost of $O(|H|)$ since we are using once the protocol 3.6.1. In the fourth step we perform a number of matrix multiplication proofs equal to the number of components in the Wedderburn decomposition. Such matrices have entries in $\mathbb{F}_{q^{k_i}}$ and dimension n_i. But by theorem 4.5.3

$$\sum_{i=1}^{t} n_i^2 k_i = |G|$$

Hence the total cost is linear in $O(|H|)$. □

We would like to underline that we proved that the existence of a transitive action of H on $X = \{1, \ldots, n\}$ is a necessary condition (Theorem 4.7.2). Hence to use the approach described here, G must contain a subgroup of order n and hence n divides $|G|$.

4.8 Final considerations

Let us summarize the main features of all the solutions that we provide:

Cut and Choose based: It is a Σ-protocol, with communication cost $O(n^2)$ and cheating probability $\frac{1}{2^n}$. It works for any permutation in S_n.

Permutation matrix based: It is a Σ-protocol, with communication cost $O(n^2)$ and cheating probability $\frac{1}{p}$. It works for any permutation in S_n.

Wedderburn based: It is a public coin relaxed Σ-protocol, with communication cost $O(n)$ and cheating probability negligible in ℓ. It works for any permutation in a group H of order n.

Generalized Wedderburn based: It is a public coin relaxed Σ-protocol, with communication cost of $O(|H|)$ and cheating probability negligible in ℓ. It works for any permutation in a group H of order m.

The permutation matrix based protocol is asymptotically better than Cut and Choose. In fact, we can require p to be very large and this makes the commitment scheme more secure and the cheating probability smaller in the permutation matrix based protocol, while it only affects the on commitment security in the other case.
The advantage of the Wedderburn based protocol should be clear: it requires a smaller communication cost, even if at the expense of the number of permutations that can be used.
Finally the generalization is helpful for groups of cardinality between n and n^2 since, beyond this bound, it loses the advantage on the communication cost that it has with respect to the permutation matrix protocol.

Bibliography

[1] I. Damgård and J.B. Nielsen. *Commitment Schemes and Zero-Knowledge Protocols*. Aarhus Univeristy, BRICS

[2] I. Damgård. *On Σ-protocols*. CPT 2010, v.2

[3] S. Lang. *Algebra*. Springer, 1993

[4] R. Cramer and I. Damgård. *Zero-knowledge for finite field arithmetic. Or: Can zero-knowledge be for free?*. In CRYPTO 1998, volume 1462 of Lecture Notes in Computer Science, pages 424-441. Springer-Verlag, 1998.

[5] R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, Universiteit van Amsterdam, Netherlands, 1997.

[6] S. de Hoogh, B. Schoenmakers, B. Škoríc, and José Villegas. *Verifiable rotation of homomorphic encryptions*. In Public Key Cryptography(2009) 393-410

[7] D. Chaum. *Untraceable electronic mail, return addresses, and digital pseudonyms*. Communications of the ACM, 24(2):84-88, 1981.

[8] K. Sako and J. Killian. *Receipt-free mix-type voting scheme*. In EUROCRYPT 1995, volume 921 of Lecture Notes in Computer Science, pages 393-403. Springer-Verlag, 1995.

[9] M. Jakobsson and A. Juels. *Mix and match: Secure function evaluation via ciphertexts*. In ASI-ACRYPT 2000, volume 1976 of Lecture Notes in Computer Science, pages 162-177. Springer-Verlag, 2000.

[10] O. Goldreich, *Foundations of Cryptography*. Cambrige University Press, 2001.

[11] V.Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambrige University Press, 2005.

[12] M. Bellare,S. Micali and R. Ostovsky, *The (True) complexity of Statistical Zero-Knowledge*. In ACM Symposium on Theory of Computing, pages 494-502, 1990

[13] R. Ostrovsky, R. Venkatesan and M. Yung. *Interactive Hashing Simplifies Zero-Knowledge Protocol Design*. In T. Helleseth (Ed.): Advances in Cryptology-Proceedings of EUROCRYPT 1993, LNCS 765, pages 267-273. Springer- Verlag, 1993.

[14] `http://www.gap-system.org/Manuals/pkg/wedderga/doc/chap0.html`

[15] C.Crépeau. *Efficient Cryptographic Protocols Based on Noisy Channels*. Advances in Cryptology: EUROCRYPT '97, volume 1233 of Lecture Notes in Computer Science, pp.110-123. Springer-Verlag 1997.

[16] C.Crépeau, J. van de Graaf and A.Tapp. *Committed Oblivious Transfert and Private Multi-Party Computations*. Advances in Cryptology: Proceeding of Crypto '95, August 1995, pp.110-123.

[17] C.Crépeau and J.Kilian. *Achieving oblivious transfert using weakened security assumptions*. In 29^{th} Symposium on Foundations of Computer Science, pages 42-52. IEEE, 1998

[18] T. P. Pedersen, *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, p.129-140, August 11-15, 1991

[19] T. Okamoto. *Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes*. In CRYPTO 1992, volume 740 of Lecture Notes in Computer Science, pages 31-53. Springer-Verlag, 1993.

[20] R.Cramer, I. Damgård, B. Schoenmakers. *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*. In CRYPTO 1994, volume 839 of Lecture Notes in Computer Science, pages 174-187. Springer-Verlag, 1994.