Selected Topics in Cryptography: from the basics to e-voting

September 20, 2022

Contents

Abstract

The purpose of these notes is to provide a self-contained introduction to relevant concepts in cryptography. We will present a toy e-voting system as as a motivational example to introduce several crypto primitives needed to implement it. In particular, we will touch basic number theory, encryption, digital signatures, hash functions, commitments, secret sharing, proof systems, zero-knowledge, and secure function evaluation.

1 Preliminaries

Notation. We use $\mathbb{N}$

to denote the set of all natural numbers. For any $n \geq 0$, we denote by $\{0,1\}^n$ the set of all binary strings of length n ($\{0,1\}^0$ is the empty set). We denote by $\{0,1\}^\star$ the union of all sets $\{0,1\}^n$ for every $n \geq 0$. We assume $\{0,1\}^\star$ to be the domain and range of any algorithm. For any natural number m, we let U_m stand for the uniform distribution over binary strings of length m. We denote by $[n]$ the set of numbers $\{1,\ldots,n\}$, by $|x|$ the bit length of $x \in \{0,1\}^\star$ and by $x||y$ the concatenation of any two strings x and y in $\{0,1\}^\star$. With a slight abuse of notation, we will sometimes assume the existence of a special symbol $\perp$ that does not belong to $\{0,1\}^\star$.

1.1 Basic discrete probability

In this section, we will give a self-contained introduction to the basic discrete probability concepts that will be needed in the rest of the notes. We remark that the topics we cover are tailored for our treatment. For instance, we will be limited to probability distributions over finite sets (not just over countable sets).

Consider an experiment that can produce a finite number of outcomes (the results of the experiments). The set of all possible outcomes of the experiments is called the *sample space* of the experiment and is often denoted as Ω. The *power set* of the sample space is the set consisting of all different sets in the sample space, that is all possible outcomes. An event is an element of the power set of the sample space that may include more than one outcome. For example, rolling a dice with 6 faces can produce 6 possible results. We can consider the experiment of rolling a dice 2 times. Thus, the subset $E \triangleq \{1, 3\}$ is an element of the power set of the sample space $\{1, 1\}, \{2, 1\}, \ldots, \{6, 6\}$ consisting of two dice rolls (the sample space in this case has 6^2 elements) and as such E is an event of the sample space. In this case E is a called an elementary event since $E \in \Omega$. When $E \in 2^\Omega$ and $E \notin \Omega$, E is called a non-elementary or compound event. A probability space is a finite sample space associated with a function that assigns to each element in the power set of the sample space a *probability*, that is a real number between 0 and 1.

Precisely we have the following formal definitions.

Definition 1 (Power set) *Let Ω be a finite set. The* power set *of Ω is denoted by 2^Ω and is defined to be the set consisting of all subsets $E \subseteq \Omega$.*

Observe that 2^Ω also includes the empty set and Ω itself and that the cardinality of 2^Ω is $2^{|\Omega|}$.

Definition 2 *A sample space Ω is a finite set. The elements of Ω are called the outcomes (or elementary events) of the sample space.*

Definition 3 *An event E of a sample space Ω is a an element of 2^Ω.*

Definition 4 *Two events E and F of a sample space Ω are* disjoint *if $E \cap F = \emptyset$.*

It is easy to see that from the properties of probability space, we have the following.

Fact 1 *If two events E and F of the same sample space are disjoint then* $\text{Prob}[E \cup F] = \text{Prob}[E] + \text{Prob}[F]$.

Definition 5 *A probability space consists of a sample space Ω and a function (called the* probability distribution*)* $\text{Prob} : 2^\Omega \to \mathbf{R}$ *called the* probability function *such that the following three properties hold:*

- $0 \leq \text{Prob}[\omega] \leq 1$ *for all $\omega \in \Omega$.*
- $\sum_{\omega \in \Omega} \text{Prob}[\omega] = 1$.
- *(This third property above could be derived from the other two, but for simplicity we include it.)*

  For any sequence of disjoint events $E_1, \ldots, E_n \in 2^\Omega, \text{Prob}[\cup_{i=1}^n E_i] = \sum_{i=1}^n \text{Prob}[E_i]$.

In this case, we will sometimes say that Prob *is a probability distribution (or just a distribution) over the finite set* Ω.

For any event $E \in \Omega$ *we call* $\text{Prob}[E]$ *the* probability *of the event* E.

The probability of a non-elementary event E of a sample space Ω can be seen to be $\text{Prob}[E] \stackrel{\triangle}{=} \sum_{\omega \in E} \text{Prob}[\omega]$, and can be seen that $\text{Prob}[\emptyset] = 0$ and $\text{Prob}[\Omega] = 1$.

Definition 6 *An event that is non-elementary is also called a* compound *event.*

Example 1 *The previous sample space consisting of rolling a dice twice can be cast in the theoretical framework of probability theory as follows. The sample space* Ω *consists of all pairs* (a, b) *such that* $a, b \in [6]$. *Moreover, we can assume that each elementary event* $E \in \Omega$ *has same probability to occur, that is the probability that any outcome will occur is the same. In this case, our probability space consists of* Ω *and the probability function* Prob *that assigns probability* $1/36$ *to each event* $E \in \Omega$. *It is easy to see that the properties of probability space are satisfied.*

Consider the event E *consisting of all rice rolls in which the first roll is* 3. *That is,* $E \stackrel{\triangle}{=} \{\{3,1\},\{3,2\},\{3,3\},\{3,4\},\{3,5\},\{3,6\}\}$ *The probability of* E *is the sum of the probabilities of all* $\omega \in E$ *and thus it equals* $1/6$.

Definition 7 (Uniform distribution) *The uniform distribution over a finite set* S *is the probability distribution with sample space* S *that assigns to each string* $x \in S$ *the probability* $\frac{1}{|S|}$.

The uniform distribution U_m *is the uniform distribution over the set* $\{0,1\}^m$ *for an integer* $m > 0$.

It is easy to see that for any finite set S *the uniform distribution over* S *induces a probability space in the obvious way.*

Two events are independent if the probability of the occurrence of both events is the product of their individual probabilities of occurrence. Precisely, we have the following.

Definition 8 (Independent events) *Two events* E *and* F *are* independent *if:*

$$\text{Prob}[E \cap F] = \text{Prob}[E] \cdot \text{Prob}[F].$$

Intuitively, the conditional probability of F given E is the probability that F occurs given that E occurred. We have the following definition.

Definition 9 (Conditional probability) *Let* E *and* F *two events of the same probability space. The* conditional probability *of* F *given* E *is denoted by* $\text{Prob}[F|E]$ *and is defined as:*

$$\text{Prob}[F|E] \stackrel{\triangle}{=} \frac{\text{Prob}[E \cap F]}{\text{Prob}[E]}.$$

It can be showed that that this induces a new probability space in which the sample space is E *instead of* Ω.

We leave as exercise to prove the following fact.

Fact 2 (Union bound) *Let $E_1, \ldots, E_n$ a sequence of events of the same probability space. Then,* $\text{Prob}[\cup_{i\in[n]} E_i] \leq \sum_{i\in[n]} \text{Prob}[E_i]$.

Exercise 1 *Prove Fact 2.*

Usually, in mathematics random variables are functions from a sample space to the real numbers. In the context of cryptography it is more useful to refer to random variables as functions mapping a sample space to a finite set, usually a set of binary strings of a certain length.

Definition 10 (Random variable) *A* random variable *associated with a probability space* (Ω, Prob) *and ranging over a finite set S is a function $X : \Omega \to S$.*

A random variable over binary strings is a random variable associated with some probability space $(\{0,1\}^n, \text{Prob})$ *and ranging over the set $\{0,1\}^m$, for some integers $n, m > 0$.*

Sometimes, when it is clear from the context we neglect the dependency from the probability space and we simply talk about a random variable over a finite set.

Definition 11 *Let X be a random variable associated with a probability space* (Ω, Prob) *and ranging over binary strings of length m, for some inter $m > 0$. Then, for each $x \in \{0,1\}^m$ we denote by* $\text{Prob}[X = x]$ *the following quantity:* $\text{Prob}[X^{-1}(x)]$.

Definition 12 *We say that a random variable X is uniformly distributed over a finite set S (or simply that X is the random variable over S) if X is a random variable associated with the probability space* $(\Omega \stackrel{\triangle}{=} S, \text{Prob})$ *and ranging over S such that X maps each string $x \in \Omega$ into itself and* Prob *assigns probability* $\frac{1}{|S|}$ *to each $x \in \Omega$ (in other words,* Prob *is uniformly distributed).*

We say that a random variable X is uniformly distributed over strings of length $m > 0$ (or simply that X is the random variable over strings of length m) if X is uniformly distributed over $\{0,1\}^m$.

Definition 13 *Let $m > 0$ be an integer and let D be a probability distribution over sample space $\{0,1\}^m$. Let $n > 0$ be an integer and $A : \{0,1\}^m \to \{0,1\}^n$ be a function. We denote by $A(D)$ the random variable associated with probability space* $(\{0,1\}^m, D)$ *and ranging over $\{0,1\}^n$ given by the composition of A and D.*

We are interested in analyzing the probability that a certain randomized algorithm A outputs a certain string. A probabilistic algorithm is an algorithm that starts with an input x and with an additional random string r that is selected from the uniform distribution of strings of a certain length $m > 0$. and during its execution can use the random bits of r to its like. Here m is some

bound on the maximum number number of random bits A will need during its execution.[1]

Definition 14 *The output of A on an input x and random string r is denoted by $A(x, r)$.*

For a given input x, we denote by $A(x, U_m)$ the random variable associated with the uniform distribution over strings of length m and ranging over strings of length m, as in Definition 13. Then, $\text{Prob}[A(x, U_m) = y]$ means the probability that algorithm A outputs y on input x. This can be seen to be equal $\frac{d}{2^m}$ where d is the set of all strings $r \in \{0,1\}^m$ such that $A(x, r) = y$. Alternatively, we use $A(x)$ to refer to the random variable $A(x, U_m)$.

Example 2 *Consider the following problem. You are given a polynomial $p(\cdot)$ of degree $n > 0$ over a field F as a list of n coefficients and you want to know if p is the 0 polynomial, i.e., the polynomial that is 0 on all inputs.*

For concreteness, we can consider the field F consisting of 2^m elements, for some large integer $m > 0$.[2] Moreover, let us suppose we can encode any field element of F into a bit string of m elements, thus we can suppose that each coefficient of $p(\cdot)$ is a string in $\{0,1\}^m$.

This problem is easy to solve in time n by checking the list of all coefficients. Now consider the following variant. You are given access to an algorithm (called the oracle*)[3] $O(\cdot)$ that allows you to evaluate the polynomial on any point, i.e., for any $d \in F, O(d)$ returns $p(d)$ and the execution of the oracle takes one step. We want to design a probabilistic algorithm with access to $O(\cdot)$ that runs in $O(1)$ steps.*

A receives as input a polynomial $p(\cdot)$ represented as a list of n coefficients in $\{0,1\}^m$, for some $n > 0$ and a random string $r \in \{0,1\}^m$ as additional random string. A interprets r as field element of F and invokes $O(r)$ to get output y and returns 1 iff $y = 0$.

Let f be the predicate defined so that for each polynomial $p(\cdot)$, $f(p) = 1$ iff p is the 0 polynomial. We are interested in analyzing the probability that A on input p outputs the correct result $f(p)$, that is $\text{Prob}[A(p, U_m) = f(p)]$.

We have two cases. If p is the 0 polynomial then, it is clear that A always outputs 0 on any string $r \in \{0,1\}^m$, so $\text{Prob}[A(p, U_m) = f(p) = 1] = 1$.

If p is not the 0 polynomial then, from basic algebra, we know that a polynomial of degree n over any field has at most n roots. So $A(p, r) = 1$ only if r is one of the n roots of p. Equivalently, $A(p, r) = 0$ for all $r \in \{0,1\}^m$ except when r is one of the $\leq n$ roots of $p(\cdot)$ So, in this case $\text{Prob}[A(p, U_m) = f(p) = 0] \geq 1 - \frac{n}{2^m}$.

Therefore, if m is large then A outputs the correct result w.v.h.p.

[1] As it will be explained later, m should be actually be a function of the length of the inputs but here for simplicity we will skip this detail; you can think of A as being an algorithm that is executed only on strings of the same length.

[2] From abstract algebra, we know that there are finite fields of size $k = p^m$ for any prime number p and integer $m > 0$.

[3] You can think of the oracle as an external library that the programmer invokes by sending a request to a server. We assume that the server magically executes the code in one step.

Definition 15 *Let X be a random variable associated with a probability space (Ω, Prob) and ranging over binary strings of length $m > 0$. For any function $f : \{0,1\}^m \to \{0,1\}^\star$, we denote by $\text{Prob}[f(x) = y|\ x \leftarrow X]$ the following quantity: $\sum_{\omega \in E} \text{Prob}[\omega]$, where $E \stackrel{\triangle}{=} \{\omega \in \Omega|\ f(X(\omega)) = y\}$. Intuitively, this is the probability that f outputs y on input x when x is* sampled *from X.*

Notice that f may also be a Boolean expression involving the variable x, see for instance Thm. 2.

Moreover, if S is a finite and $f : S \to \{0,1\}^\star$ is a function, we define $\text{Prob}[f(x) = y|\ x \leftarrow S]$ in the following way. Let X_S the random variable that is uniformly distributed over S and let (Ω, Prob) be the corresponding probability space. Then, $\text{Prob}[f(x) = y|\ x \leftarrow S]$ is the following quantity: $\sum_{\omega \in E} \text{Prob}[\omega]$, where $E \stackrel{\triangle}{=} \{\omega \in \Omega|\ f(X(\omega)) = y\}$.

Definition 16 *Expected value of a random variable] Let X be a random variable ranging over a finite set of real numbers (or a finite set of binary strings that represent integers). The* expected value *(or* mean*) of X is the quantity $E(X)$ defined in the following way:*

$$E(X) \stackrel{\triangle}{=} \sum_{x \in S} x \cdot \text{Prob}[X = x].$$

Example 3 *Let $\Omega \stackrel{\triangle}{=} \{1,2,3,4\}$ be a sample space and let* Prob *be defined so that $\text{Prob}[\{1\}] = \frac{1}{3}, \text{Prob}[\{2\}] = \frac{1}{6}, \text{Prob}[\{3\}] = \frac{1}{6}, \text{Prob}[\{4\}] = \frac{1}{3}$. Let X be the random variable associated with probability space (Ω, Prob) that maps $x \in \Omega$ into x^2.*

Then $E(X) = \text{Prob}[X = 1] \cdot 1 + \text{Prob}[X = 4] \cdot 4 + \text{Prob}[X = 9] \cdot 9 = \frac{1}{3} + \frac{1}{6} \cdot 4 + \frac{1}{6} \cdot 9 + \frac{1}{3} \cdot 16 = \frac{5}{2} + \frac{16}{3}$.

Example 4 *Let us compute the expected value of the random variable $A(p, U_m)$ where A is the algorithm of Example 2 and $p(\cdot)$ is a non-zero polynomial with exactly n roots.*

By the definition of expected value and from the fact that A outputs a binary value, it holds that $E(X) = 0 \cdot \text{Prob}[A(p, U_m) = 0] + 1 \cdot \text{Prob}[A(p, U_m) = 1] = \text{Prob}[A(p, U_m) = 1] = \frac{n}{2^m}$.

Birthday paradox. What is the probability that among n persons in a room at least two of them share the same birthday supposing that the people are randomly chosen)? The solution to this question is analyzed and generalized by the following theorem in which B can be thought as the number of days in a year.

Theorem 1 *Let B be an integer > 0 and let n be an integer > 0. Let $S \stackrel{\triangle}{=} \{(x_1, \ldots, x_n)|\ x_1, \ldots, x_n \in [B]\}$.*

If $n \geq \sqrt{2} \cdot B^{1/2}$ then $\text{Prob}[\exists i \neq j \in [n]$ s.t. $r_i = r_j|\ (r_1, \ldots, r_n) \leftarrow S] \geq 1 - \frac{1}{e} > 0.6 calc.$[4]

[4] Cfr. this notation with Def. 15.

Proof 1 *We have the following.*

$$\begin{gathered}
\text{Prob}[\exists i \neq j \in [n] \text{ s.t. } r_i = r_j |\ (r_1, \ldots, r_n) \leftarrow S] = \\
1 - \text{Prob}[\forall i \neq j\ r_i \neq r_j |\ (r_1, \ldots, r_n) \leftarrow S] = \\
1 - \frac{(B-1)}{B} \cdot \frac{(B-2)}{B} \cdots \frac{(B-n+1)}{B} = \\
= 1 - \prod_{i=1}^{n-1} (1 - \frac{i}{B}) \geq {}^{5} \\
= 1 - \prod_{i=1}^{n-1} e^{-\frac{i}{B}} = \\
= 1 - e^{-\frac{1}{B}\sum_{i=1}^{n-1} i} \geq \\
1 - e^{-\frac{n^2}{2B}}.
\end{gathered}$$

Then,if $n \geq \sqrt{2} \cdot B^{1/2}$ *then* $1 - e^{-\frac{n^2}{2B}} \geq 1 - \frac{1}{e}$ *and thus* $\text{Prob}[\exists i \neq j \in [n] \text{ s.t. } r_i = r_j |\ (r_1, \ldots, r_n) \leftarrow S] > 1 - \frac{1}{e}$*, as we had to prove.*

Exercise 2 *Use Thm. to find the minimum integer* n *such that a group of* n *randomly chosen people, at least* 2 *of them will share the same birthday with probability* > 0.6*.*

See also Exercises 6 and 10.

Definition 17 (Computational problems with verifiable solutions) *A computational problem* P *with* unique solution *is formulated in this way.* P *is associated with a finite set* S *of pairs* (x, s) *where* x *is the public input of the problem and* s *is the secret, and a function* $f : S \to \{0,1\}^\star$ *that transforms such pairs into the* instance *to the problem. Moreover, there exists an algorithm* V *such that for any* $(x, s) \in S), V(x, f(x, s), s') = 1$ *iff* $s' = s$*.*

We say that an algorithm A *solves* P *with probability* p *if:*

$$\text{Prob}[V(A(x, f(x))) = 1 |\ (x, s) \leftarrow S] = p.$$

Fact 3 *Let* P *be a problem with verifiable solutions. Suppose we have an algorithm* A *that is able to find a solution to* P *with probability* $\frac{1}{\lambda}^d$ *for some* $d > 0$*.*

Then, A *can be used to build the following algorithm* B*.* B*, given input* (x, y)*, runs* $A(x, y)$ *on* λ^d *independent executions to attempt to find a solution in one of the* λ^d *executions and outputs the first solution found. Precisely,* B *uses the output* y *of* A *in any of the* d *executions as input to* V *to check if* y *is a solution to the problem.*

[5] Here, we apply the inequality $1 - x \leq e^{-x}$ that follows from the Taylor's expansion $e^{-x} = 1 - x + \frac{x^2}{2} + \ldots$.

First, observe that the probability that B does not find a solution in any of the d executions $(1 - \frac{1}{\lambda}^d)^{\lambda^d}$.

Recall that the inverse of the Neper number e^{-1} *can be approximated by the following formula:* $(1 - \frac{1}{n})^n \approx e^{-1}$. *Therefore, it is easy to see that the probability that B finds a solution is* $1 - e^{-1}$.

1.2 An asymptotic treatment of crypto

Modern cryptography is based on the hypothesis there are some mathematical problems that are deemed to be *hard* to solve for modern computers with limited resources. In theory, the problems on which modern encryption schemes are based can be broken by computers running with enough time, but there is no known way to break these problems in, let us say, 100 years with the fastest supercomputer on the earth. Therefore, modern crypto is based on the assumptions that (1) the security has to be preserved only against adversaries running in a reasonable amount of time and (2) that adversaries breaking a system with a very small probability are not considered a threat.

We need to be precise on how (1) and (2) are quantified.

In crypto and generally in computer science, the notion of efficient algorithm is captured by *probabilistic polynomial-time* (PPT) algorithms.

An algorithm A is *probabilistic* (or randomized) if A, during its execution, can choose random bits (also called coin tosses). Recall that an algorithm A takes as input a bit string x of finite length, and for sake of simplicity an input to an algorithm is often represented as a tuple of inputs $(x_1, \ldots, ms, x_m)$ for some integer $m > 0$.

Recall that if A is a probabilistic algorithm, then $A(x_1, x_2, \ldots, x_m)$ denotes the random variable of the output of A when A is run on input $(x_1, x_2, \ldots, x_m)$ and a sufficiently long random string. Instead, $A(x_1, x_2, \ldots, x_m; r)$ denotes the output of A when run on input $(x_1, x_2, \ldots, x_,)$ and (sufficiently long) random string r. An algorithm A is polynomial-time (PT) if there exists a polynomial $p(\cdot)$ such that for any input $x \in \{0,1\}$ of length n, A on input x runs in at most $p(n)$ steps. That is, the execution of A is bounded by a polynomial number of steps in the length of the input. Therefore, when we say that an algorithm is *efficient* we will mean that it is a PPT algorithm.

The notion of PPT is implicitly asymptotic. The running time of the algorithm is a function of the length of the input. This is the standard approach used in computer science and in particular in complexity theory. Likewise running times, in order to analyze the probability that an algorithm breaks a cryptosystem or a problem, we will also consider the probability of breaking such a system or problem as a function of some parameter. Usually, the parameter we will consider is the *security parameter*. When the parties setup the cryptosystem, they choose a concrete security parameter λ. Both the running times of the parties and the probability of adversaries of breaking the system will be seen as function of this parameter. The idea is that longer is the security parameter, smaller should be the probability that the crypto-system or problem can be broken. The analysis will tell us that this probability is asymptotically

small. This is of course a theoretical guarantee but provides a sound and solid framework to analyze security of systems.

The notation of small probability used in crypto is the following. A function of the security parameter (representing the probability that an attacker breaks a system or problem for varying lengths of the security parameter) is considered small if it is smaller than the inverse of any polynomial. Precisely, we have the following definition of *negligible function*.

Definition 18 (negligible function) *A* negligible *function* $\mathsf{negl}()$ *is a function with domain the positive integers and range the real numbers that is smaller than the inverse of any polynomial in* λ *(starting from a certain point). Formally, a function* f *with domain the positive integers and range the real numbers is negligible if for any* $d \geq 0$*, there exists* $\lambda_0 > 0$ *such that for any* $\lambda \geq \lambda_0$*,* $f(\lambda) < \frac{1}{\lambda^d}$*.*

For instance, the function $\frac{1}{2^\lambda}$ is a negligible function. The function $\frac{1}{\lambda^2}$ is not negligible since it is not asymptotically smaller than, e.g., $\frac{1}{\lambda^3}$.

Why do we consider negligible functions as synonym of small probability and not for instance functions like $\frac{1}{\lambda}$? That is, why does an adversary that breaks a crypto-system with probability $\frac{1}{\lambda}^2$ is still considered a threat even if these functions happen to be very small for sufficiently long values of λ? (Here, by breaking a crypto-system we mean breaking a computational problem with verifiable solutions. Cf. Def. 17.)

The reason is the following. Suppose we have an adversary A that is able to break a crypto-system with probability $\frac{1}{\lambda}^d$ for some $d > 0$.

Then, by fact 3, it can be seen that the probability of breaking the crypto-system can be amplified to $1 - e^{-1}$, that in practice is considered a very high probability of succeeding in breaking the system. Therefore, adversaries that break crypto-systems with probabilities like $\frac{1}{\lambda^d}$ are a threat because such probabilities of success can be *amplified* to high probabilities. If instead the probability of an adversary in breaking the system is $\frac{1}{2^\lambda}$, we are safe because, by fact 3, it would be necessary to run A 2^λ times to make this probability high, but $> 2^\lambda$ steps are not feasible for any computer in the universe.

A stronger model of computation is the one of *non-uniform algorithms*. We will use it only in the negative way, that is for saying that even such an algorithm cannot solve a problem or break a crypto-system. Therefore, if a crypto-system cannot be broken by a non-uniform algorithm it is stronger guarantee than the fact that such system cannot be broken by a standard algorithm. A non-uniform algorithm $A \stackrel{\triangle}{=} \{A_n\}_{n>0}$ is a family of algorithms. Intuitively, each algorithm A_n in the family is supposed to work only on instances of length n. So, while a (uniform) algorithm has to work on inputs of any length, a non-uniform algorithm consists of an infinite series of algorithms, each of which runs only on instances of some specified length. This is a more powerful model of computation as shown in the next example.

Example 5 (The power of non-uniformity) *Let L be a subset of $\{0,1\}^\star$; subsets of $\{0,1\}^\star$ are called* languages *(see Section 1.3). We say that a possibly unbounded algorithm A decides a language L if: for every $x \in L, A(x) = 1$ and for every $x \notin L, A(x) = 0$.*

We then define the halting language L^H *to be the set of all binary strings representing algorithms (computer programs) that halt on a fixed input, for instance the bit 0.*

It is known that the halting language is undecidable*, that is there is no (possibly unbounded) algorithm that decides L^H.*

Consider now the following language L^1: a binary string $x \in L^1$ iff there exists an integer $n > 0$ such that[6] $x = 1^n$ and the encoding of the integer n as bit string represents a computer program that halts.

It is possible to prove that L^1 is undecidable iff L^H is undecidable, so L^1 is undecidable.

Now consider the following non-uniform algorithm $A = \{A_n\}_{n>0}$. For each $n > 0$ we construct A_n as follows. If there exists a string $x_n \in L^1$ such that $|x_n| = n$, then A_n works as follows: A_n on input a string x outputs 1 iff $x = x_n$. If there is no $x_n \in L^1$ such that $|x_n| = n$, then A_n outputs 0 on any input.

It is easy to see that A decides L^1 in the following sense: for each $x \in L^1$, $A_{|x|}(x) = 1$ and for each $x \notin L$, $A_{|x|}(x) = 0$. So, L^1 is undecidable by (uniform) algorithms but can be decided by non-uniform algorithms and this demonstrates that non-uniform algorithms are strictly more powerful than (uniform) algorithms.

We now formalize the notion of non-uniform efficient algorithms, that is non-uniform PPT algorithms.

Definition 19 (non-uniform PPT algorithm) *A non-uniform PPT (nuPPT) algorithm A is a family of probabilistic algorithms $\{\mathsf{A}_\lambda\}_{\lambda>0}$ parameterized by λ such that there exists a polynomial $p(\cdot)$ such that for all $\lambda > 0$, A_λ takes as inputs strings of length at most $p(\lambda)$, runs in time at most $p(\lambda)$ and its description has length at most $p(\lambda)$.*

Observe that in the previous exercise the non-uniform algorithm that decides L^1 is actually a nuPPT algorithm as well, so we have an example of a computational task that is undecidable by unbounded power (uniform) algorithms but that is easy for nuPPT algorithms.

The notion of efficient computation leads to another natural notion of equivalency of objects. Two objects are computationally identical if no efficient procedure can differentiate them. We consider this notion for probabilistic objects, in particular for random variables. The notion is formalized by considering families of random variables indexed by a parameter λ. Intuitively, two sequences of random variables $\{X_\lambda\}_{\lambda>0}$ and $\{Y_\lambda\}_{\lambda>0}$ are computationally indistinguishable if no efficient algorithm can distinguish if a sample was sampled by X_λ or Y_λ for

[6] Here, 1^n is the binary string consisting of the bit 1 repeated n times.

all sufficiently large values of λ and with all except with probability negligible in λ.

Precisely, given two families of random variables $X_0 \stackrel{\triangle}{=} \{X_{0,\lambda}\}_\lambda$ and $X_1 \stackrel{\triangle}{=} \{X_{1,\lambda}\}$ over binary strings, a function $\epsilon(\cdot)$ and a nuPPT algorithm $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda>0}$, we say that $\mathcal{D}$ *distinguishes* X_0 from X_1 with *advantage* $\epsilon(\cdot)$ whether $|\text{Prob}[\mathcal{D}_\lambda(x) = 1|\ x \leftarrow X_{0,\lambda}] - \text{Prob}[\mathcal{D}_\lambda(x) = 1|\ x \leftarrow X_{1,\lambda}]| = \epsilon(\lambda)$.

Definition 20 (Computationally indistinguishable random variables) *We say that two families of random variables $X_0 = \{X_{0,\lambda}\}_\lambda$ and $X_1 = \{X_{1,\lambda}\}$ are computationally indistinguishable if for every nuPPT distinguisher $\mathcal{D}$, $\mathcal{D}$ distinguishes X_0 fronm X_1 with advantage $\epsilon(\cdot)$, for some negligible function $\epsilon(\cdot)$.*

In Thm. 6 we will show that the notion of computational indistinguishability is transitive, i.e., if X, Y, Z are families of random variables and X and Y are computationally indistinguishable and Y and Z are computationally indistinguishable, then X and Z are computationally indistinguishable.

Notice that the notion of computational indistingushability is a weakening of statistical indistinguishability that we now recall.

Definition 21 (Statistical distance and ϵ-closeness) *Let D_1 and D_2 be two probability distributions over a finite set S. Their statistical distance (SD) is*

$$|D_1 - D_2| \stackrel{\triangle}{=} \frac{1}{2}\sum_{s\in S} |D_1(s) - D_2(s)|.$$

If $|D_1 - D_2| \leq \epsilon$, we say that D_1 and D_2 are ϵ-close. Analogously, let X_1 and X_2 be two random variables over a finite set S (the union of the ranges of X_1 and X_2). Their statistical distance (SD) is

$$|X_1 - X_2| \stackrel{\triangle}{=} \frac{1}{2}\sum_{s\in S} |\text{Prob}[X_1 = s] - \text{Prob}[X_2 = s]|.$$

If $|X_1 - X_2| \leq \epsilon$, we say that X_1 and X_2 are ϵ-close.

Note that two distributions are identically distributed when they are 0-close.

Two families of random variables indexed by λ are statistically indistinguishable if their SD is a negligible function when viewed as function of λ. Precisely, we have the following definition.

Definition 22 (Statistically indistinguishable random variables) *We say that two families of random variables $X_0 = \{X_{0,\lambda}\}_\lambda$ and $X_1 = \{X_{1,\lambda}\}$ are statistically indistinguishable if for any $d > 0$, there exists an integer $\lambda_0 > 0$ such that for every $\lambda \geq \lambda_0$, $|X_{0,\lambda} - X_{1,\lambda}| < \frac{1}{\lambda^d}$.*

From the definition of SD, it is straightforward to see the following.

Let X be a random variable over a set S. Let f be any function with domain S and range S'. We denote by $f(X)$ the random variable that assigns to each element $s' \in S'$ probability $\sum_{s\in f^{-1}(s')} \text{Prob}[X = s]$. Alternatively, $f(X)(s') \stackrel{\triangle}{=} \text{Prob}[X = f^{-1}(s')]$.

Fact 4 *For any set S, any function f over domain S and any two ϵ-close random variables X_1 and X_2 over S, it holds that $|f(X_1) - f(X_2)| \leq \epsilon$.*

The proof can be found at [`https://www.ccs.neu.edu/home/wichs/class/crypto-fall15/lecture2.pdf`, Theorem 4]. The previous fact implies that if two random variables are ϵ-close then the distinguishing advantage of of any distinguisher, even of unbounded power, that attempts to distinguish the two random variables will be at most ϵ.

Fact 5 *If two families of random variables $X_0 = \{X_{0,\lambda}\}_\lambda$ and $X_1 = \{X_{1,\lambda}\}_\lambda$ are $\epsilon(\cdot)$-close, (i.e. for every λ, $X_{0,\lambda}$ is $\epsilon(\lambda)$-close to $X_{1,\lambda}$), there is no non-uniform distinguisher $\mathcal{D}$ (even computationally unbounded) that can distinguish X_0 from X_1 with advantage more than $\epsilon(\cdot)$.*

As corollary, we have the following equivalent definition of statistical indistinguishability.

Definition 23 (Equivalent definition of statistically indistinguishable random variables) *We say that two families of random variables $X_0 = \{X_{0,\lambda}\}_\lambda$ and $X_1 = \{X_{1,\lambda}\}$ are statistically indistinguishable if there exists no non-uniform distinguisher (even computationally unbounded) $\mathcal{D}$ and no non-negligible function $\epsilon(\cdot)$ such $\mathcal{D}$ distinguishes X_0 from X_1 with advantage more than $\epsilon(\cdot)$.*

1.3 Languages and decision problems

A language is a (possibly infinite) subset of $\{0,1\}^\star$. The notion of languages is rooted in complexity theory and is needed to formalize hard *decision* problems. In fact, many problems in computer science can be cast as deciding if a particular input belongs or not to a specific sets. As an example, circuit satisfiability is the problem of deciding if a given input represents a Boolean circuit that is satisfiable (i.e., such that there exists an input for which the circuit outputs 1). Such problem can be formalized by defining the language L of circuit satisfiability as the set of all strings representing a Boolean circuit that is satisfiable and then the corresponding decision problem is the problem of deciding if a string $x \in L$ or $x \notin L$. If $x \in L$ we say that x is an *instance* of the language. Intuitively, an algorithm decides a language L if it can tell if any string $x \in L$ or $x \notin L$. This notion can be generalized to deciding languages in probability (that is, the algorithm can err in deciding if a string belongs or not to the language=. Precisely, we have the following definitions.

Definition 24 ((p,q)-decidability) *Let p, q be functions with domain the positive integers and range $[0,1]$. (p and q can be seen as the probabilities of resp. erring in deciding when $x \in L$ (resp. $x \notin L$) with prob. p (resp. q). That is, p is the completeness (resp. soundness) error.) A language L is (p,q)-decidable by a PPT algorithm A if the following holds:*

- Completeness. *if $x \in L$ then* $\text{Prob}[A(x) = 1] \geq p(|x|)$.

- Soundness. *There exists an integer* $n_0 > 0$ *such that for every* $x \notin L, |x| \geq n_0$ *then* $\text{Prob}[A(x) = 1] \leq q(|x|)$.

Let C *be a class of algorithm (i.e., a subset of all possible algorithms). We also say that membership in* L *can be* (p, q)*-decided by algorithms in* C*. If* p *and* q *are resp. the constant functions* 1 *and* 0*, we say that membership in* L *can be decided by algorithms in* C*.*

The notion of PT algorithms is equivalent to the set of all algorithms belonging to the following P.

Definition 25 (P-language) *A language* L *is in* P *if membership in* L *can be decided by a (deterministic) polynomial-time algorithm.*

Since efficiency is synonym of PPT algorithms, we will consider a language to be *hard* if such language is not decidable by PPT algorithms. This is formalized as follows.

Definition 26 (BPP-language) *A language* L *is in the class (bounded-error probabilistic polynomial time)* BPP *if membership in* L *can be* $(\frac{2}{3}, \frac{1}{3})$*-decided by a PPT algorithm.*

The choice of the constants is arbitrary. The error probability does not even have to be constant: the same class of problems is defined by allowing error as high as $\frac{1}{2} - n^{-c}$ on the one hand, or requiring error as small as 2^{-nc} on the other hand, where c is any positive constant, and n is the length of input. This flexibility in the choice of error probability is based on the idea of running an error-prone algorithm many times, and using the majority result of the runs to obtain a more accurate algorithm.

Definition 27 (hard language) *A language* L *is* hard *if* $L \notin \mathsf{BPP}$*.*

We will now recall some standard notions of complexity theory that are fundamental in the treatment of zero-knowledge proof systems that we will consider soon. All practical claims one wants to prove with cryptographic proof systems can be described as statements concerning membership in a particular NP language. While ¶ is the class of easy languages NP is the class of languages that are easy to verify, that is for which membership in the language can be decided with the help of a *witness*. One example is the language of circuit satisfiability. Given a Boolean circuit that is satisfiable, an input that satisfies the circuit is a witness that proves that the circuit is member of the language.

In applications, the prover will own this additional information, the witness, while the verifier will not. The following notion of *polynomial-time relations* formalizes this asymmetry of information: in the context of proof systems that we will see later, a pair (x, w) belonging to the relation consists of the common input x among the prover and the verifier and the witness w that is known only to the prover.

Definition 28 (Polynomial-time relation) *A* polynomial-time *relation $\mathcal{R}$ is a relation over pairs of binary strings for which membership of (x, w) in $\mathcal{R}$ can be decided by a deterministic polynomial-time algorithm and there exists a polynomial p such that for each $(x, w) \in \mathcal{R}$, $|w| \leq p(|x|)$.*

If $(x, w) \in \mathcal{R}$ then we say that w is a witness *for* instance *(or statement) x. In this case, we also write $\mathcal{R}(x, w) = 1$. The instance can be thought as a computational problem and the witness as a solution to the problem.*

Definition 29 (NP-language) *A language L is in* NP *if there exists a polynomial-time relation $\mathcal{R}$ such that:*

$$x \in L \iff \exists w \in \{0,1\}^\star \ \mathcal{R}(x, w) = 1.$$

[7]

There are some languages in NP that are harder to decide than others. In particular, the theory of NP-hardness tells us that there are some problems that are as hard as any other problem in NP. This is formalized as follows.

Definition 30 (Poly-time reducibility) *We say that a language L_1 is poly-time reducible to a language L_2 if there exists a function f over domain and range $\{0,1\}^\star$ such that:*

- *f is polynomial-time computable.*
- *For any $x \in \{0,1\}^\star, x \in L_1 \iff f(x) \in L_2$*

If L_1 is poly-time reducible to L_2 we write $L_1 \leq_p L_2$

Definition 31 (NP-hard and NP-complete languages) *A language L is* NP*-hard if for any* NP*-language L_2, $L_2 \leq_p L$. An* NP*-complete language is a language that is both in* NP *and* NP*-hard.*

Note that if $L_1 \leq_p L_2$ and $L_2 \in \mathsf{P}$ (i.e., L_2 can be decided in (deterministic) polynomial-time) then L_1 can be also decided in polynomial-time with the following algorithm: Given a string x, compute $x' = f(x)$ and then decide whether $x' \in L2$. Similarly, if $L_1 \leq_p L_2$ and $L_2 \in \mathsf{NP}$ then $L_1 \in \mathsf{NP}$ as well.

In the context of proof systems, we will be interested only in NP-hard languages. The reason is that if a language is not hard, the verifier does not need a proof from the prover of the fact that some statement belongs or not to the language.

A polynomial-time relation $\mathcal{R}$ is naturally associated with the NP language $L_\mathcal{R}$ defined as $L_\mathcal{R} = \{x \mid \exists w : \ (x, w) \in \mathcal{R}\}$. Similarly, an NP language is naturally associated with a polynomial-time relation. Given an NP language L, for any natural number $k > 0$, we denote by L_k the language $L \cap \{0,1\}^{\leq k}$.

[7] Note that, by our definition of polynomial-time relation, w has length polynomial in $|x|$.

2 Public-key Encryption

Motivation. The basic idea of our toy e-voting scheme will be to transmit the voters' preference in some encrypted form so as to preserve the privacy of the individual voters but still allowing to compute the result of the election. To this purpose, we introduce the concept of public-key encryption.

Public-key encryption (PKE) is one of the most basic crypto primitives. PKE allows Alice to send an encrypted message to Bob, a *ciphertext*. Alice encrypts her message under the *public-key* (PK) of Bob and Bob can decrypt the ciphertext using the corresponding *secret key* (SK). Intuitively, only Bob should be able to decrypt and a malicious Eve who intercepts the ciphertext should not be able to leak any information about the encrypted message (except trivial information like the length of the encrypted message). The latter property is equivalent to say that no attacker, given the PK, can distinguish with non-negligible probability whether a ciphertext encrypts one of two messages of its choice. This is formalized as follows.

Definition 32 (Public-key Encryption scheme) *A public-key encryption (PKE) scheme* $\mathsf{E} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *over message space* $\mathsf{M} = \{M_\lambda\}_{\lambda>0}$ *is a tuple of* 3 *PPT algorithms.*

- $\mathsf{Setup}(1^\lambda)$*: on input the security parameter* λ *output* public key PK *and* secret key SK*.*
- $\mathsf{Enc}(\mathsf{PK}, M)$*: on input public key* PK *and a message* $M \in M_\lambda$*, outputs* ciphertext CT*.*
- $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})$*: on input the secret key* SK *for security parameter* λ *and a ciphertext* CT *output a message* $M \in M_\lambda$ *or* $\perp$*.*

A PKE scheme has to satisfy the following properties:

- (perfect) correctness*, that is that for all* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda)$*, all* $M \in M_\lambda$*, for all* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, M)$*,* $\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}) = M$*.*

  (In a PKE scheme with statistical correctness, the previous property would hold in probability over the random coins used to encrypt.)

- Indistinguishability against chosen message attacks ($\mathsf{IND\text{-}CPA}$)*: for every* 2 *families of messages* $M \stackrel{\triangle}{=} \{M_\lambda\}_{\lambda>0}, M' \stackrel{\triangle}{=} \{M'_\lambda\}_{\lambda>0}$ *such that for any* $\lambda > 0, M_\lambda, M'_\lambda \in M_\lambda$ *and* $|M_\lambda| = |M'_\lambda|$*, and every nuPPT adversary* $\mathsf{A} = \{\mathsf{A}_\lambda\}_{\lambda>0}$*, the following function is negligible in* λ*:*

$$\begin{aligned}&\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,M'}(\lambda) \stackrel{\triangle}{=} \\ &\big|\mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK}, \mathsf{Enc}(\mathsf{PK}, M_\lambda)) = 1 \mid \mathsf{PK} \leftarrow \mathsf{Setup}(1^\lambda))] - \\ &\quad \mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK}, \mathsf{Enc}(\mathsf{PK}, M'_\lambda)) = 1 \mid \mathsf{PK} \leftarrow \mathsf{Setup}(1^\lambda))].\end{aligned}$$

  The function $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,M'}(\cdot)$ *is called the* advantage *of* A *in breaking the* $\mathsf{IND\text{-}CPA}$ *property against the PKE scheme* E *for (families of) messages*

M and M'. Sometimes, we will abuse the notation considering M and M' as single messages m, m' and not families of messages. In this case, the implicit families of messages M and M' are the families in which for each value of λ, $M_\lambda = m$ and $M'_\lambda = m'$.

Observe that the property of IND-CPA is formulated by means of the *indistinguishability paradigm*. That is, we state that no efficient adversary can distinguish, except with negligible advantage, among two possible worlds.

Exercise 3 *Suppose you have a PKE scheme E such that a ciphertext of E encrypting a plaintext M reveals $f(M)$ for some "non-trivial function" f. Show a specific non-trivial function f such that in this case E would not be* IND-CPA*. Moreover, discuss what are the trivial functions such that E in that case would still be* IND-CPA*.*

Exercise 4 *Prove or refuse the following claim: no PKE scheme can hide the length of the encrypted plaintext, i.e., a ciphertext always reveals the length of the plaintext.*

Exercise 5 (intermediate difficulty) *Suppose you have both a PKE scheme E and a symmetric-key encryption scheme E'. Let us fix a specific security parameter λ. Suppose that in E a ciphertext encrypting a string of n bits with a PK for parameter λ has size $O(n^2)$ but in E' a ciphertext encrypting a string of n bits with a secret-key for parameter λ has size $O(n)$ bits. Show how to combine E and E' to build a new PKE scheme E'' that is more efficient in terms of bandwidth, that is such that the length of a ciphertext encrypting a string of n bits with a PK for parameter λ has length $O(n)$ bits.*

The need for computationl assumptions for achieving PKE. In order to PKE to exist, one needs to assume that some computational assumptions hold. First, notice that a PKE scheme cannot be secure against adversaries with unbounded power: an adversary, given the PK, can just search by brute force over all random coins of the setup algorithm until it finds the corresponding SK and breaks the system. Observe also that the assumption that $\mathsf{P} \neq \mathsf{NP}$ is also a *necessary* assumption for PKE to exist. In fact, the following problem can be seen to be in NP: given a PK PK, decide if a given bit of the SK SK equals 1. So, if $\mathsf{P} = \mathsf{NP}$, the SK can be computed from the PK and the PKE can be broken. This is a necessary assumption but not sufficient (that is, we do not have proofs that if $\mathsf{P} \neq \mathsf{NP}$ then PKE exists. We will soon see other computational assumptions on which PKE can be based.

Roadmap for PKE. In these notes, we will consider the *ElGamal encryption scheme* as one candidate for PKE. ElGamal is based on algebraic groups for which some computational assumptions hold. So, we will follow an abstract approach. In Section 2.1 We will first present the notion of *secure groups* (that is, algebraic groups of cryptographic interest), that is generic algebraic groups

on which some hard problems can be formulated. In Section 2.2 we will implement ElGamal from such generic groups. Finally, in Section 2.3 we will present a concrete algebraic group on which such hard problems are conjectured to hold. Note that in Section 2.1 we will present some hard problems that can be formulated in *any* algebraic group of prime order. So, given a concrete group, these problems may or not may hold in that group. So the assumptions stated therein are *generic* in the sense that they do not refer to a concrete group. Only in Section 2.3 such assumptions will be stated in a concrete group.

2.1 Secure Algebraic Groups

We assume the reader to be familiar with the notion of algebraic group (henceforth, just a group) that we will recall for sake of completeness.

Definition 33 (Group) *A group $\mathbb{G} = (S, \cdot)$ consists of a set S and an operation $\cdot : (S \times S) \to S$ such that the following properties hold.*

- *Associativity. For any $g, h, u \in S$, it holds that $(g \cdot h) \cdot u = g \cdot (h \cdot u)$.*
- *Existence of identity. There exists a distinct element $1 \in S$, called the identity, such that for any $g \in S$ it holds that $g \cdot 1 = g$.*
- *Existence of inverses. For any $g \in S$ there exists $h \in S$ such that $g \cdot h = 1$; such element will be denoted by g^{-1}.*
- *Commutativity. For any $g, h \in S$ it holds that $g \cdot h = h \cdot g$. (In some texts, this property is not part of the definition of groups and groups satisfying this additional properties are called* abelian groups*. Our definition of group thus corresponds to the one of abelian groups.*

A group $(S, \cdot)$ is *finite* when the set S is finite. Henceforth, a group will be synonym of a finite group. Moreover, with a slight abuse of notation if $\mathbb{G} = (S, \cdot)$ is a group we will write $a \in \mathbb{G}$ to say that $a \in S$ and when it is clear from the context we will sometimes write ab instead of $a \cdot b$; so, we will often say that $\mathbb{G}$ is a group omitting to specify its set and operation.

Definition 34 (Subgroup) *A subset S of a group $\mathbb{G}$ is a* subgroup *of $\mathbb{G}$ if S is closed under the operation of the group, that is for each $x, y \in S$, $x \cdot y \in S$.*

An example of finite group is the following.

Fact 6 *The set $\mathbb{Z}_p^\star$ consisting of the integers $1, \ldots, p-1$ is a group with identity 1 with the group operation being the multiplication modulo p.*

Definition 35 (Group order) *The order of a group $(S, \cdot)$ is the cardinality of the set S.*

Fact 7 *The identity element in a group is unique.*

Proof 2 *Let $\mathbb{G}$ be a group. If e, f are identity elements of $\mathbb{G}$ then $e = e \cdot f = f$.*

It is also see to check the following fact.

Fact 8 *The inverses of elements in a group are unique.*

In order to state computational assumptions on groups we will need to treat groups asymptotically and moreover to assume that these groups are somehow efficient. Moreover, for simplicity we will just focus on groups of prime order.

Theorem 2 *Let $\mathbb{G}$ be a group of order N. Then for any $g \in \mathbb{G}$, it holds that $g^N = 1$.*

Proof 3 *First, observe that for any $g \in \mathbb{G}$, it holds that for any two different $a, b \in \mathbb{G}$, $g \cdot a \neq g \cdot b$. Indeed, suppose towards a contradiction that $g \cdot a = g \cdot b$. Then $g^{-1} \cdot g \cdot = g^{-1} \cdot g \cdot b$ and, by properties of the inverses, this implies $a = b$, contradicting the fact that $a \neq b$.*

Therefore, if $g_1, \ldots, g_N$ are all elements of $\mathbb{G}$ then $(g \cdot g_1), \ldots, (g \cdots g_N)$ are also all elements of $\mathbb{G}$. Let h be $\prod_{i=1}^{N} g_i$.

We have the following.

$$h \stackrel{\triangle}{=} \prod_{i=1}^{N} g_i = \prod_{i=1}^{N} (g \cdot g_i) = g^N \cdot \prod_{i=1}^{N} g_i = g^N \cdot h.$$

Hence, by multiplying both sides by h^{-1} we have that $g^N = 1$, as we had to show.

It is easy to check that the previous theorem implies the following corollary.

Theorem 3 *If $\mathbb{G}$ is a finite group of order N, for any $g \in \mathbb{G}$ and any integer $d \geq 0$, then $g^d = g^{d \mod N}$.*

Theorem 4 *In a group $\mathbb{G}$ of prime order, any element of $\mathbb{G}$ is a generator of $\mathbb{G}$.*

Proof 4 *Let p be the order of $\mathbb{G}$ and let $g, h \in \mathbb{G}$.*

Consider the set A of all distinct powers of g: $g^0 = 1, g^1, \ldots$. Since the group is finite, then A is finite as well and let d be its cardinality. Clearly, $d \leq p - 1$. If $d = p - 1$, then all powers of g are distinct elements of $\mathbb{G}$ and so there exists $i \leq d$ such that $h = g^i$, as we had to prove. Suppose towards a contradiction that $d < p - 1$. There, there exists the smallest integer $p - 1 > d \geq 0$ such that $g^0, \ldots, g^d$ are distinct $g^{d+1} = g^j$ for $j \leq d$. Since the inverse of g^j can be seen to be g^{p-j}, then we have that $g^{d+1+p-j} = 1$, so $g^{d+1-j} = 1 = g^0$. So there exists an integer $p - 1 > m \geq 1$ such that $g^m = 1$ contradicting the uniqueness of the identity.

Theorem 2 implies the following corollary, known as the *Fermat's little theorem.*

Fact 9 *For each integer a in $\mathbb{Z}_p^\star$, $g^{p-1} = 1$.*

Definition 36 (Generator of a family of groups) *A PPT algorithm* Gen *is called a generator of a family of groups if the following properties hold.*

- *Efficient generation of the group description.* Gen*, on input the security parameter λ, outputs a description of a group $\mathbb{G}_\lambda$ along with the prime number q_λ of $\lambda + 1$ bits and a generator g_λ of this group.*
- *Efficient operations. Group operations for groups output by* Gen *are efficient relative to the security parameter λ.*
- *Efficient decidability. Membership in any group $\mathbb{G}$ output by* Gen *can be tested efficiently.*
- *Efficient samplability. Elements can be sampled efficiently from any group output by* Gen*.*

We can see that Gen *can be used to generate a family of groups $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda>0}$ such that for each $\lambda > 0$, $\mathbb{G}_\lambda$ consists of q_λ elements and q_λ is prime of $\lambda + 1$ bits. However, notice that the output of* Gen *is not a fixed family of groups but it is random variable over them.*

Definition 37 (generator of a group) *Let $\mathbb{G}$ be a (finite) group of order N. An element $a \in \mathbb{G}$ is a* generator *of $\mathbb{G}$ if for any $h \in \mathbb{G}$ there exists an integer $i \geq 0$ such that $h = g^i$.*

We are now ready to state a computational assumption, called the *Decisional Diffie-Hellman Assumption* (DDH) on which the security of ElGamal can be based. Observe that we state the assumption with respect to a generator of a family of groups (that is, for any generator of family of groups Gen we have the DDH assumption relative to Gen). Later, we will show concrete groups on which such assumption is believed to hold and so we will talk about the DDH assumption in a specific family of groups.

2.1.1 The DLOG and DDH assumptions

Assumption 1 (DDH Assumption over Gen**)** *Let* Gen *be a generator of a family of groups. Let $X_{0,\lambda}$ be the random variable $(\mathbb{G}_\lambda, q_\lambda, g, h, u, v)$, with $(\mathbb{G}_\lambda, q_\lambda, g) \leftarrow \mathsf{Gen}(1^\lambda)$, $w \leftarrow \mathbb{Z}_{q_\lambda}^\star, g, h \leftarrow \mathbb{G}_\lambda, u \stackrel{\triangle}{=} g^w, v \stackrel{\triangle}{=} h^w$. Let $X_{1,\lambda}$ be the random variable $(\mathbb{G}_\lambda, q_\lambda, g, h, u, v)$, with $(\mathbb{G}_\lambda, q_\lambda) \leftarrow \mathsf{Gen}(1^\lambda)$, $w \leftarrow \mathbb{Z}_{q_\lambda}^\star, g, h \leftarrow \mathbb{G}_\lambda, u \stackrel{\triangle}{=} g^w, v \leftarrow \mathbb{G}_\lambda$. We say that the Decisional Diffie-Hellman assumption (DDH) holds for* Gen *if for every nuPPT algorithm $\mathsf{A} = \{\mathsf{A}_\lambda\}_{\lambda>0}$, the following function is negligible in λ:*

$$\left|\mathrm{Prob}[\mathsf{A}_\lambda(x) = 1|\ x \leftarrow X_{0,\lambda}] - \mathrm{Prob}[\mathsf{A}_\lambda(x) = 1|\ x \leftarrow X_{1,\lambda}]\right|.$$

The previous probability is called the advantage of A in breaking DDH (over Gen*) and is denoted by $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{DDH},\mathsf{Gen}}(\cdot)$.*

DDH holds if it holds over some generator of family of groups Gen.

We call the family of random variables $\{X_{0,\lambda}\}_{\lambda>0}$ *(resp.* $\{X_{1,\lambda}\}_{\lambda>0}$*) the family of valid DDH tuples (resp. random DDH tuples).*

Essentially, the DDH assumption states that an attacker, given g, g^a, g^b, cannot distinguish whether an element Z equals $g^{a \cdot b}$ or is randomly chosen in the group generated by g.

The security of ElGamal will be based on the DDH assumption. However, it is instructive to present a weaker and more basic computational assumption, namely the *discrete log assumption*.

Assumption 2 (Discrete Log Assumption over Gen**)** *Let* Gen *be a generator of a family of groups. Let* X_λ *be the random variable* $(\mathbb{G}_\lambda, q_\lambda, g, h, w)$*, with* $(\mathbb{G}_\lambda, q_\lambda, g) \leftarrow \mathsf{Gen}(1^\lambda)$*,* $w \leftarrow \mathbb{Z}^\star_{q_\lambda}, h \stackrel{\triangle}{=} g^w$*. We say that the discrete log assumption (DLOG) holds for* Gen *if for every nuPPT algorithm* $\mathsf{A} = \{\mathsf{A}_\lambda\}_{\lambda>0}$*, the following quantity is negligible in* λ*:*

$$\mathrm{Prob}[\mathsf{A}_\lambda(\mathbb{G}_\lambda, q_\lambda, g, h) = w |\ (\mathbb{G}_\lambda, q_\lambda, g, h, w) \leftarrow X_\lambda].$$

The previous probability is called the probability that A *breaks DLOG (over* Gen*).*

DLOG holds if it holds over some generator of family of groups Gen.

It is straightforward to see that if DDH holds over some generator Gen then DLOG holds over Gen. Indeed, if there existed an adversary that, given g, u computed as in both real or random DDH tuples, is able to compute w, then such an adversaty could be used to distinguish real from random DDH tuples. In this sense, the DLOG is weaker than the DDH assumption. In particular, there could exist a generator Gen for a family of groups such that the DLOG assumption holds over Gen but DDH does not hold over Gen. Unfortunately, we will be able to prove the security of ElGamal only under the DDH assumption.

Exercise 6 *Charlie claims to have found a group of prime order* B*, with* B *that is approximatively* 2^{128}*, and he claims that no supercomputer in the world is currently able to solve instances of the discrete logarithm problem in this group.*

Can we refuse the claim of Charlie just knowing the order of the group and even without looking at the technical details of the group? The answer is surprisingly yes.

Consider the following attacker A *against the discrete logarithm problem over a generic group* $\mathbb{G}$ *of order* B*. By generic here we mean that the attacker works against any group, precisely the attacker will use just the group operation to perform the attack and will not use any particular inner detail of the group.*[8]

A *takes as input an instance of the discrete log problem, precisely the description of a group* $\mathbb{G}$ *of order* B*, the group order* B*, a generator* g *of the group* $\mathbb{G}$ *and an element* $h \in \mathbb{G}$ *that equals* g^w *for some random* $w \in \mathbb{Z}_B$*.*

[8] Formally speaking, a generic attacker could be defined as an algorithm that has access to an oracle that implements the group operation and tests equality among group elements. For simplicity, we do not do that and We skip details.

A *chooses many random values* $x_1, \ldots, x_n, z_1, \ldots, z_n \in [B]$ *for some* n *to be defined later.*

A *computes the two sets* $S_1 \triangleq \{g^{x_1}, \ldots, \ldots, g^{x_n}\}$ *and* $S_2 \triangleq \{h \cdot g^{z_1}, \ldots, \ldots, g^{z_n}\}$.

Now observe that each two elements of S_1 *are different w.v.h.p. Similarly, each two elements of* S_2 *are different w.v.h.p. as well.*

Then, if $S_1 \cap S_2 \neq \emptyset$*, it must be that there exists* $i, j \in [B]$ *such that* $g^{x_i} = h \cdot g^{z_j}$*. Then, it is easy to see that* $x_i - y_j = w \mod B$*, and so the solution can be found if the event* $S_1 \cap S_2 \neq \emptyset$ *occurs.*

(Notice that A *takes as input the description of the group but it is using it only to perform group operations and is not carrying out any particular computation based on the inner structure of the group.)*

Use Thm. 2 to find the parameter n *so that* A *finds a solution to any instance of the discrete logarithm problem over the Charlie's group in a reasonable amount of time (for instance less than one year with a supercomputer) with probability* > 0.6.

2.2 ElGamal encryption

We will now present the ElGamal encryption scheme. Actually, we will present what other authors call the *exponential ElGamal* PKE. In standard ElGamal, the message space consists of group elements. Instead in the variant we will present, the message space consists of integers in the exponent. That is, we will encrypt an integer M by actually encrypting g^M and then decryption will work by decrypting the group element g^M and by finding (via brute force) the integer M. In order for this brute force procedure to succeed we indeed need to assume that the message space is small. This will be useful to use ElGamal to build our toy e-voting scheme.

We present ElGamal with respect to a generic generator of family of groups. That is, the presentation is abstract and does not depend on a particular group.

Definition 38 (ElGamal) *Let* Gen *be a generator of a family of groups. Let* $\mathsf{M} \triangleq \{M_\lambda\}_{\lambda>0}$ *be a family of sets in which each* $M_\lambda, \lambda > 0$ *is a subset of* $\{0,1\}^\lambda$ *of polynomial cardinality in* λ*. Our ElGamal encryption scheme* $\mathsf{ElGamal} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *over message space* M *is a tuple of* 3 *PPT algorithms.*

$\mathsf{Setup}(1^\lambda)$*: on input the security parameter* λ *it runs* $(\mathbb{G}_\lambda, q_\lambda, g) \leftarrow \mathsf{Gen}(1^\lambda)$*, computes* $h = g^w$ *for randomly chosen integer in* $\mathbb{Z}^\star_{q_\lambda}$ *and outputs PK* $\mathsf{PK} \triangleq (\mathbb{G}_\lambda, q_\lambda, g, h)$ *and SK* $\mathsf{SK} \triangleq w$*. For simplicity, we will omit* $\mathbb{G}_\lambda$ *and* q_λ *from the description of the PK.*

$\mathsf{Enc}(\mathsf{PK}, M)$*: on input public key* $\mathsf{PK} \triangleq (g, h)$ *and a message* $M \in M_\lambda$ *(parsed as an integer in* $\mathbb{Z}_{q_\lambda}$*, outputs* ciphertext $\mathsf{CT} \triangleq (g^r, h^r \cdot g^M)$*.*

$\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})$*: on input secret key* $\mathsf{SK} \triangleq w$ *and ciphertext* $\mathsf{CT} \triangleq (\mathsf{CT}_1, \mathsf{CT}_2)$*, compute* $y = \mathsf{CT}_2 \cdot \mathsf{CT}_1^{-w}$ *and, by brute force search over all elements*

$M \in \mathsf{M}$ *until an element* M *such that* $y = g^M$ *is found and in such case output* M*; if no such element can be found, output* $\perp$*. (Notice that, by our assumption on the cardinality of* $M_\lambda, \lambda > 0$*, this brute force computation can be done efficiently.)*

When it is clear from the context, we sometimes assume Dec *to just output* y*, that is* Dec *computes* y *as above and outputs it.*

It is easy to see that the following facts hold.

Fact 10 $\mathsf{ElGamal}$ *satisfies perfect correctness as in Def. 32.*

Exercise 7 *In ElGamal the message space consists of group elements. Show how to modify ElGamal so as to encrypt arbitrary bit strings. Hint: use hash functions (see later sections for reference).*

Exercise 8 *In this exercise we consider the version of ElGamal for arbitrary message space developed at the previous exercise. Observe that if a ciphertext* CT *of ElGamal for PK* PK *(corresponding to SK* SK*) encrypts a plaintext* M*, decrypting* CT *with a wrong SK* $\mathsf{SK}' \neq \mathsf{SK}$ *always results in an output* M'*. Show how to modify ElGamal so that when you decrypt an ElGamal ciphertext with a wrong SK results in an error* $\perp$*; the resulting scheme may satisfy statistical correctness. Hint: restrict the message space.*

2.2.1 Security of ElGamal under the DDH Assumption

We will now analyze the security of the ElGamal PKE scheme. As we said, PKE cannot be based on information theoretically arguments, that is we cannot prove that there exist PKE schemes secure against adversaries of unbounded power. Moreover, we are currently not able to prove the security of PKE schemes without assuming that some mathematical conjectures hold, and in particular that some computational assumptions hold.

The security of crypto protocols follows the *reduction* paradigm. That is, one proves the security of, e.g., a PKE scheme as follows. Suppose towards a contradiction that there exists some efficient adversary that breaks the scheme. Then, such an adversary can be used to build another adversary that breaks some computational problem that is conjectured to be hard. In this case, we say that the scheme is secure under such computational problem. This will be the case of the ElGamal scheme under the DDH assumption.

Theorem 5 *If DDH holds over generator* Gen*, then* $\mathsf{ElGamal}$ *is* $\mathsf{IND\text{-}CPA}$

Proof 5 *We first prove that (1) for any three messages* M_0, M_1, M_2*, for any* $\lambda > 0$ *if* $\mathsf{Advg}^{\mathsf{E},M_0,M_2}_{\mathsf{A}_\lambda}(\lambda) = \epsilon$*, for some* $\epsilon(\cdot)$*, then either* $\mathsf{Advg}^{\mathsf{E},M_0,M_1}_{\mathsf{A}_\lambda}(\lambda)$ *or*

$\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_1,M_2}(\lambda)$ *is* $\geq \epsilon/2$. *This follows from the following equations.*

$$\epsilon \leq \mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_0,M_2}(\lambda) \stackrel{\triangle}{=}$$

$$|\mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_0))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))] - \mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_2))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))]| =$$

$$|(\mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_0))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))] - \mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_1))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))])+$$

$$(\mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_1))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))] - \mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_2))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))])| =$$

$$|\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_0,M_1}(\lambda) + \mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_1,M_2}(\lambda)| \leq$$

$$|\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_0,M_1}(\lambda)| + |\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_1,M_2}(\lambda)|.$$

Therefore, $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_0,M_1}(\lambda)$ *and* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M_1,M_2}(\lambda)$ *cannot be both* $< \epsilon/2$.

The previous fact also implies the following fact (2): for any pair of messages M, M', *for any* $\lambda > 0$ *if* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,M'}(\lambda) = \epsilon$, *for some* ϵ, *then if* R *is randomly and uniformly sampled from any subset of the message space, either* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,R}(\lambda)$ *or* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},R,M'}(\lambda)$ *is* $\geq \epsilon/2$.

Suppose towards a contradiction that there exists a PPT adversary A *and two family of messages* $M \stackrel{\triangle}{=} \{M_\lambda\}_{\lambda>0}, M' \stackrel{\triangle}{=} \{M'_\lambda\}_{\lambda>0}$ *such that* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,M'}(\lambda) = \epsilon(\lambda)$, *for some non-negligible function* $\epsilon(\cdot)$ *of* λ.

By fact (2), for any $\lambda > 0$, *if* R *is randomly and uniformly sampled from* $\{0,1\}^\lambda$, *either* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,R}(\lambda)$ *or* $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},R,M'}(\lambda)$ *is* $\geq \epsilon(\lambda)/2$.

Suppose w.l.o.g that $\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},M,R}(\lambda) \geq \epsilon(\lambda)/2$, *where* R *is randomly sampled as above. This means that* $|p_1(\lambda) - p_2(\lambda)| \geq \epsilon(\lambda)/2$, *where*

$$p_1(\lambda) \stackrel{\triangle}{=} \mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))],$$

and

$$p_2(\lambda) \stackrel{\triangle}{=} \mathrm{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},R)=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda)), R\leftarrow\{0,1\}^\lambda]|.$$

Now consider the following algorithm B *against DDH.*

B, *on input the security parameter* 1^λ, *receives as input a tuple* $X_\lambda = (q_\lambda, g, h, u, v)$, *sets* $\mathsf{PK} = (g, h)$, $\mathsf{CT}_1 = u, \mathsf{CT}_2 = v \cdot g^{M_\lambda}$, *and outputs* $\mathsf{A}(\mathsf{PK}, (\mathsf{CT}_1, \mathsf{CT}_2))$.

Notice that if X_λ *comes from the family of valid DDH tuples, then* $v = h^w$, *where* w *is s.t.* $u = g^w$. *It is easy to see that* (u, v) *is distributed identically to* $\mathsf{Enc}(\mathsf{PK}, 1_\lambda)$, *where* 1_λ *is the encoding of the identity of the group* $\mathbb{Z}_{q_\lambda}$ *and so* $(u, v \cdot g^{M_\lambda})$ *is distributed identically to* $\mathsf{Enc}(\mathsf{PK}, M_\lambda)$. *Therefore, the probability that* B *outputs* 1 *is exactly* $p_1(\lambda)$.

On the other hand, if X_λ comes from the family of random DDH tuples, then $v = g^z$, for a random $z \in \mathbb{Z}^\star_{q_\lambda}$, but g^z is distributed identically to $h^w \cdot g^z$, where w is such that $u = g^w$ and z is random in $\mathbb{Z}^\star_{q_\lambda}$, and so (u, v) is distributed identically to $\mathsf{Enc}(\mathsf{PK}, R)$. Therefore, also $(u, v \cdot g^{M_\lambda})$ is distributed identically to $\mathsf{Enc}(\mathsf{PK}, R)$, and thus the probability that B outputs 1 is exactly $p_2(\lambda)$.

Therefore, B breaks DDH with advantage $\geq \epsilon(\lambda)/2$, a non-negligible probability, contradicting the fact that DDH holds over Gen.

The previous proof outlines a common technique used in cryptography called the *hybrid method*. This is a generalization of the fact that computationally indistinguishable random variables are transitive. We formalize it by a means of the following theorem.

Theorem 6 *Let $X_1, \ldots, X_m$ be a sequence of random variables over bit strings of finite length. Assume that a (possibly unbounded) distinguisher $\mathcal{D}$ distinguishes X_1 from X_m with advantage $> \epsilon \geq 0$. Then there exists some $i \in [m-1]$ s.t. $\mathcal{D}$ distinguishes X_i from X_{i+1} with advantage $> \frac{\epsilon}{m}$.*

Proof 6 *Suppose $\mathcal{D}$ distinguishes X_1 from X_m with advantage $> \epsilon$. That means that $|\mathrm{Prob}[\mathcal{D}(x) = 1 \mid x \leftarrow X_1] - \mathrm{Prob}[\mathcal{D}(x) = 1 \mid x \leftarrow X_m]| > \epsilon$. Let $s_i = \mathrm{Prob}[\mathcal{D}(x) = 1 \mid x \leftarrow X_i]$, $i \in [m-1]$.*

Thus, $|s_1 - s_m| > \epsilon$. This implies:

$$\begin{aligned} \epsilon &< |s_1 - s_m| \\ &= |s_1 - s_2 + s_2 - s_3 + \cdots + s_{m-1} - s_m| \\ &\leq |s_1 - s_2| + |s_2 - s_3| + \cdots + |s_{m-1} - s_m|. \end{aligned}$$

Therefore, there must exist $i \in [m-1]$ such that $|s_i - s_{i+1}| > \frac{\epsilon}{m-1} \geq \frac{\epsilon}{m}$.

The previous theorem is easily extended to families of random variables in the obvious way and in that form is used in security proofs in the following way. When trying to prove that two families of random variables $X \stackrel{\triangle}{=} \{X_\lambda\}_{\lambda>0}$ and $Y \stackrel{\triangle}{=} \{Y_\lambda\}_{\lambda>0}$ are indistinguishable, one can define intermediate *hybrid (family of) random variables* $X_1, \ldots, X_m$ such that $X_1 = X$ and $Y = X_m$ and prove that two generic hybrids are distinguishable with advantage at most $\epsilon(\cdot)$. Then, it follows that no distinguisher can distinguish X from Y with advantage more than $\epsilon(\cdot)/m$.

Usually, in the security proof $\epsilon(\cdot)$ represents a negligible function. Moreover, m can be also replaced by a function of the security parameter. In that case, the theorem guarantees that, until the number of hybrids is polynomial in the security parameter and any two pairs of hybrids are distinguishable with at most negligible advantage, then X and Y cannot be distinguished with more than negligible advantage as well.

We now show another example of the hybrid methodology.

Definition 39 (Pseudorandom generator) *A pseudorandom generator (PRG) with expansion of $m > 0$ bits is a PPT algorithm* PRGEN *with domain $\{0,1\}^\star$ and range $\{0,1\}^\star$ such that for any $n > 0$ the output of* PRGEN *on input a string $x \in \{0,1\}^n$ is a string of $n + m$ bits and it holds the following* pseudorandomness *property: for any PPT distinguisher $\mathcal{D}$ there exists a negligible function $\epsilon(\cdot)$ such that for any $n > 0$,* $|\text{Prob}[\mathcal{D}(\mathsf{PRGEN}(\mathsf{k})) = 1 \mid \ \mathsf{k} \leftarrow \{0,1\}^{\mathsf{n}}] - \text{Prob}[\mathcal{D}(\mathsf{k}) = 1 \mid \ \mathsf{k} \leftarrow \{0,1\}^{\mathsf{n+m}}]| = \epsilon(\mathsf{n})$.

We denote the random variables of the strings that $\mathcal{D}$ receives as input in the left case the pseudorandom random variable $P^{n,m}$ for parameters n, m whereas we denote the one on the right the truly random variable $R^{n,m}$ for parameters n, m.

Thus, the pseudorandomness property for PRGEN can be reformulated as follows: for any any PPT distinguisher $\mathcal{D}$ there exists a negligible function $\epsilon(\cdot)$ such that for any $n > 0$, $|\text{Prob}[\mathcal{D}(k) = 1 \mid \ k \leftarrow P^{n,m}] - \text{Prob}[\mathcal{D}(k) = 1 \mid \ k \leftarrow R^{n+m}]| = \epsilon(n)$.

Given a PRG PRGEN, for any $m \geq 0$ we denote by $\mathsf{PRGEN^m}$ the following algorithm that we define inductively. For $m = 0$, $\mathsf{PRGEN^0}$ is the identity function. For $m > 0$, $\mathsf{PRGEN^m}$ is defined so that for any $n > 0$, any string $k \in \{0,1\}^n$: $\mathsf{PRGEN^m(k)} \stackrel{\triangle}{=} \mathsf{PRGEN(PRGEN^{m-1}(k))}$.

Let us assume that PRGEN is a PRG with 1-bit expansion and let m be an integer > 1. We want to prove the following.

Theorem 7 *If* PRGEN *is a PRG with 1-bit expansion, then* $\mathsf{PRGEN^m}$ *is a PRG with m-bit expansion.*

Proof 7 *Let X be the random variable of the output of* $\mathsf{PRGEN^m}$ *on input a string of n bits (actually it is a family of random variables parameterized by n). Let Y be the random variable of random strings of $n + m$ bits (another family of random variables parameterized by n).*

We want to prove that no PPT distinguisher $\mathcal{D}$ can distinguish X from Y with more than negligible advantage.

Suppose by contradiction that there exists a PPT distinguisher $\mathcal{D}$ that distinguishes X from Y with advantage $> \epsilon(\cdot)$, for some non-negligible function $\epsilon(\cdot)$. Let $X_i, i \in [m+1]$ be the hybrid random variables defined to be the output of $\mathsf{PRGEN^{m-(i-1)}(k)}$ *with $k \leftarrow \{0,1\}^{n+i-1}$. It is easy to see that X_1 is identical to X and X_{m+1} to Y.*

By Thm. 6, there exists $i \in [m]$ such that $\mathcal{D}$ that distinguishes X_i from X_{i+1} with advantage $> \epsilon(\cdot)/(m+1)$.

We use $\mathcal{D}$ to build the following PPT distinguisher $\mathcal{D}'$ that breaks the pseudorandomness property of PRGEN*: $\mathcal{D}'$ receives as input a string $k \in \{0,1\}^\star$ and outputs $\mathcal{D}(\mathsf{PRGEN^{m-i}(k)})$.*

For any $n > 0$, let us analyze the behavior of $\mathcal{D}'$ when $\mathcal{D}'$ receives an input k that is supposed to come from either the pseudorandom random variable $P^{n+i-1,1}$ or the truly random variable $R^{n+i-1,1}$ for parameters $n + i - 1, 1$. We have that:

$$|\text{Prob}[\mathcal{D}'(k) = 1 \mid \ k \leftarrow P^{n+i-1,1}] - \text{Prob}[\mathcal{D}'(k) = 1 \mid \ k \leftarrow R^{n+i-1,1}]| =$$

$$|\text{Prob}[\mathcal{D}(\mathsf{PRGEN}^{\mathsf{m}-\mathsf{i}}(\mathsf{k}))=1 \mid \ \mathsf{k}\leftarrow \mathsf{P}^{\mathsf{n}+\mathsf{i}-1,1}]-\text{Prob}[\mathcal{D}(\mathsf{PRGEN}^{\mathsf{m}-\mathsf{i}}(\mathsf{k}))=1 \mid \ \mathsf{k}\leftarrow \mathsf{R}^{\mathsf{n}+\mathsf{i}-1,1}]| =$$
$$|\text{Prob}[\mathcal{D}(k)=1 \mid \ k\leftarrow X_i]-\text{Prob}[\mathcal{D}(k)=1 \mid \ k\leftarrow X_{i+1}]| > \epsilon(n)/(m+1),$$

contradicting the pseudorandomness property of PRGEN.

Exercise 9 (advanced) *Consider the following extensionf of* IND-CPA*. A PKE scheme satisfies indistinguishability against chosen message attacks for multiple messages (*mIND-CPA*) if: for every $n > 0$, for every 2 families of tuples of messages $\vec{M} \stackrel{\triangle}{=} \{\vec{M}_\lambda\}_{\lambda>0}, \vec{M}' \stackrel{\triangle}{=} \{\vec{M}'_\lambda\}_{\lambda>0}$ such that for any $\lambda > 0, \vec{M}_\lambda, \vec{M}'_\lambda$ are tuples of n bit strings such that for any $i \in [n]$ $|M_{i,\lambda}| = |M'_{i,\lambda}|$, and every nuPPT adversary $\mathsf{A} = \{\mathsf{A}_\lambda\}_{\lambda>0}$, the following function is negligible in λ:*

$$\mathsf{Advg}_{\mathsf{A}}^{\mathsf{E},\vec{M},\vec{M}'}(\lambda) \stackrel{\triangle}{=}$$
$$\big|\text{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M_{1,\lambda}),\ldots,\mathsf{Enc}(\mathsf{PK},M_{n,\lambda}))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))]-$$
$$\text{Prob}[\mathsf{A}_\lambda(\mathsf{PK},\mathsf{Enc}(\mathsf{PK},M'_{1,\lambda}),\ldots,\mathsf{Enc}(\mathsf{PK},M'_{n,\lambda}))=1 \mid \mathsf{PK}\leftarrow\mathsf{Setup}(1^\lambda))].$$

Show that IND-CPA *is equivalent to* mIND-CPA*. Hint: use the hybrid methodology.*

2.2.2 E-voting from Homomorphic Encryption

We will now show that ElGamal (in the exponential variant we presented) satisfies nice *homomorphic* properties and we will show how to exploit them to build our toy e-voting scheme. Intuitively, given an ElGamal ciphertext CT (CT') encrypting an integer m (resp m'), we can build a ciphertext CT_h encrypting $m + m' \mod q$ (where g is the order of the group on which ElGamal is instantiated). Moreover, this can *publicly* done, that is it can be done without knowledge of the SK. Indeed, CT (resp. CT') has the form $(g^r, g^m \cdot h^r)$ (resp. $(g^{r'}, g^{m'} \cdot h^{r'})$) and so multiplying the left and right entries we can build the pair $(g^{r+r'}, g^{m+m'} \cdot h^{r+r'})$ that is a ciphertext encrypting $m + m' \mod q$ with randomness $r + r' \mod q$.

The same can be done to multiply an encrypted message by a constant.

Definition 40 (Operations on El Gamal ciphertexts)

- *Multiplication of El Gamal ciphertexts. Let $\mathsf{CT}_1 \stackrel{\triangle}{=} (\mathsf{CT}_1^l, \mathsf{CT}_1^r), \mathsf{CT}_2 \stackrel{\triangle}{=} (\mathsf{CT}_2^l, \mathsf{CT}_2^r)$ be two El Gamal ciphertexts for the same public key PK over a group of order q. We denote by $\mathsf{CT}_1 * \mathsf{CT}_2$ the ciphertext $(\mathsf{CT}_1^l \cdot \mathsf{CT}_2^l, \mathsf{CT}_2^r \cdot \mathsf{CT}_2^r)$, that is we multiply the two ciphertexts entry by entry.*

  If CT_1 encrypts message m_1 and CT_2 encrypts message m_2 then $\mathsf{CT}_1 * \mathsf{CT}_2$ encrypts message $m_1 + m_2 \mod q$.

- *Exponentiation of an El Gamal ciphertext to a constant. Let y be an integer and $\mathsf{CT} \stackrel{\triangle}{=} (\mathsf{CT}^l, \mathsf{CT}^r)$ an El Gamal ciphertext for a public key over a group of order q. We denote by CT^y the ciphertext $((\mathsf{CT}^l)^y, (\mathsf{CT}^r)^y))$, that is we exponentiate each entry of the ciphertext to y.*

  If CT encrypts message m, then CT^y encrypts message $y \cdot m \mod q$.

It is easy to see that the following facts hold.

Fact 11 *Let* $\mathsf{CT}_1\,\mathsf{CT}_2$ *be two El Gamal ciphertexts for the same public key* PK *over a group of order* q*. If* CT_1 *encrypts message* m_1 *and* CT_2 *encrypts message* m_2 *then* $\mathsf{CT}_1 * \mathsf{CT}_2$ *encrypts message* $m_1 + m_2 \mod q$*.*

Let y *be an integer and* $\mathsf{CT} \stackrel{\triangle}{=} (\mathsf{CT}^l, \mathsf{CT}^r)$ *an El Gamal ciphertext for a public key over a group of order* q*. If* CT *encrypts message* m*, then* CT^y *encrypts message* $y \cdot m \mod q$*.*

Our toy e-voting scheme: first iteration. We will construct our toy e-voting by a sequence of refinements in which, at each step, we will explain what are the issues in the previous and current refinements, and how they can be overcome, and this will motivate the introduction of new crypto primitives. As first iteration, we can imagine that each voter V encrypts his own preference using ElGamal encryption under the PK of some authority A and sends it to a public bulletin board (PBB). Let us assume that the preferences are 1 (YES) or 1 (NO), that is the election is a *referendum*. Let $\mathsf{CT}_1 \ldots, \mathsf{CT}_N$ be N ciphertexts encrypting the preferences v_i of voters $1, \ldots, N$. The authority can use the homomorphic properties of ElGamal to produce a ciphertext CT encrypting the sum of the v_i's, adds CT to PBB. Let v be the sum of v_i's. Notice that v is the result (the *tally*) of the referendum, that is the number of voters who voted YES. A can decrypt CT using its own SK to get v and announce the result of the election.

What are the issues with this scheme? First, A can use the SK not only to decrypt the ciphertext CT containing the tally v, but can decrypt each CT_i's individually to leak the preference of each voter. We will tackle this problem of *privacy* in Section 4. Another issue is the *verifiability* (or integrity) of the system. The problem is that the authority could announce a result v' different from v. Moreover, a voter could encrypt not just 0 or 1 but an arbitrary integer subverting the result of the election: for example if all voters except V vote 0 and V votes $N/2 + 1$ the result of the election will be falsely that the majority of voters voted YES. We will tackle the problem of verifiability in Section 3.

2.3 Example of a secure group: quadratic residues modulo a prime

Definition 41 (Quadratic residues modulo a prime.) *Let* p *be a prime. An integer* $y \in \mathbb{Z}_p^\star$ *is a quadratic residue (QR) modulo* p *if there exists an integer* x *such that* $y = x^2 \mod p$*. The integer* x *that verifies such equation is called a* square root *of* y *modulo* p*. The set of all quadratic residues modulo* p *is denoted by* QR_p*.*

Theorem 8 *For any prime* $p > 2$*, half of the integers in* $\mathbb{Z}_p^\star$ *are in* QR_p*.*

Proof 8 *Consider the function* f *that maps an integer* x *in* $\mathbb{Z}_p^\star$ *to* $x^2 \mod p$*. We claim that this is a 2-1 function. Indeed, let* y *be an arbitrary element of*

QR_p. *So, there exists* x *such that* $y = x^2 \mod p$. *Notice that* $(-x)^2 \mod p = x^2 \mod p = y$. *If* $-x = x \mod p$ *then, multiplying both sides by the inverse of* x, *we would have that* $1 = -1 \mod p$ *and thus* $1 = p - 1$ *over the integers, contradicting the fact that* $p > 2$. *Now, suppose towards a contradiction that there exists* $z \in \mathbb{Z}_p^\star$ *such that* $z^2 \mod p = y$ *and* $z \notin \{x, -1\}$. *Then,* $z^2 = x^2 \mod p$ *and this implies that* $(z-x)(z+x) = 0 \mod p$, *that is* $z = x$ *or* $z = -x$. *Therefore,* y *has exactly* 2 *squares and so* $|\mathsf{QR}_p| = |\mathbb{Z}_p^\star|$.

Theorem 9 QR_p *is a subgroup of* $\mathbb{Z}_p^\star$.

Proof 9 *Suppose that* $y_1, y_2 \in \mathsf{QR}_p$. *Then there exist integers* $x_1, x_2 \in \mathbb{Z}_p^\star$ *such that* $y_1 = x_1^2 \mod p, y_2 = x_2^2 \mod p$, *It is easy to observe that* $y_1 \cdot y_2$ *has* x_1, x_2 *as square root, so* $y_1 \cdot y_2 \in \mathsf{QR}_p$. *Moreover,* $1 \in \mathsf{QR}_p$ *since its square root is* 1 *itself. Therefore,* QR_p *is a subgroup of* $\mathbb{Z}_p^\star$ *as we had to prove.*

Fact 12 *If* $p = 2q + 1$, *where both* p *and* q *are odd primes, then* QR_p *is a group of prime order. An integer* p *of this form is called a* strong prime.

Proof 10 *By Theorem 8,* $|\mathsf{QR}_p| = |\mathbb{Z}_p^\star|/2 = (p-1)/2 = q$. *and by Theorem 9 it is a group.*

Why working in QR modulo a strong prime? The group QR_p where p is a strong prime is considered secure and the DDH Assumption is considered to hold with respect to generators that output such groups, that is they output the description of $\mathbb{Z}_p^\star$ for a strong prime p along with a generator of QR_p). To show that it is crucial to choose the group securely in this way, we now demonstrate that the DDH Assumption can be broken when the generator of the group generates all $\mathbb{Z}_p^\star$ and so is not a quadratic residue.

Fact 13 *An integer* $y \in \mathsf{QR}_p$ *iff* $y^{(p-1)/2.} = 1 \mod p$

If $y \in \mathsf{QR}_p$ *then let* y *be a square root of* y. *Then* $y^{(p-1)/2} = (x^2 \mod p)^{(p-1)/2.} = x^{(p-1)} \mod p$ *that, by Little's Fermat Theorem, is equal to* 1 *modulo* p. *On the other hand, suppose towards a contradiction that (1)* $y^{(p-1)/2.} = 1$ *but* $y \notin \mathsf{QR}_p$. *Let* g *be a generator of* $\mathbb{Z}_p^\star$. *Then (2)* $y = g^{2k+1}$ *for some integer* $k \geq 0$. *By (1) and (2), we have that* $1 = y^{(p-1)/2} = (g^{2k+1})^{(p-1)/2} = g^{k(p-1)} \cdot g^{(p-1)/2} = g^{(p-1)/2}$. *But* g *has order* $p - 1$, *a contradiction.*

Therefore, there exists an efficient test to decide whether an integer is QR or not. This test can be used to break the DDH Assumption with a constant advantage. Observe first that if g generates all $\mathbb{Z}_p^\star$, g has order $p - 1$, an even number.

Moreover, if $g \notin \mathsf{QR}_p$ then in a valid DDH tuple of the form (g, g^a, g^b, g^{ab}), for random $a, b \in [1, p-1]$, a and b have probability $1/2$ to be even and so g^a and g^b have probability $1/2$ to be QR modulo p. But the probability that $g^{a \cdot b \mod p-1}$ is QR modulo p is exactly the probability that $a \cdot b \mod p$ is even and so is $3/4$ (because it can be odd only when both a and b are odd).

On the other hand, in random DDH tuple of the form (g, g^a, g^b, g^z), for random $a, b, z \in [1, p-1]$, $g^z \in \mathsf{QR}_p$ with probability $1/2$ because half of the integers in the group generated by g are QR modulo p. This in turn gives an efficient procedure to break the DDH Assumption when we work in the group generated by a generator of all $\mathbb{Z}_p^\star$. Note instead that the above attack does not work when g generates a group of prime order q. Indeed, in that case $a \cdot b \mod q$ has $1/2$ probability of being even.

The group of QR modulo a strong prime also thwarts other known attacks and currently is considered pre-quantum safe.

3 Zero-knowledge Proof Systems

Motivation. The first iteration of our toy e-voting scheme suffers a major problem. The authority A can completely subvert the result of the election by announcing a different result, that is the result announced by A may not correspond to the true result. Furthermore, in a referendum voters can submit invalid preferences different from $\{0, 1\}$ with the purpose of giving more weight to one of the two options.

This problem is addressed using zero-knowledge proofs. The idea is that both the voters and A can add some proofs that prove that they made the computation honestly (the authority decrypt correctly and the voters encrypt a vote in the valid range). S It is trivial to see that these proofs exist: the voters could reveal the randomness used to encrypt and A could reveal the SK. Unluckily, these trivial proofs would reveal the individual preferences of each voters and so do not solve our problem. Zero-knowledge proofs are instead proofs that can be verified but still do not leak additional information about the statement to prove.

In particular, we will need non-interactive zero-knowledge proofs, that is proofs that are publicly verifiable by everybody, not only by whom participated in the voting process. To that purpose, we will first present interactive zero-knowledge proof systems, then we will introduce a variant of them, sigma protocols that can be turned in efficient non-interactive zero-knowledge proofs usable in our e-voting protocol.

3.1 Interactive Zero-Knowledge Proof systems

Interactive machines. The notion of interactive proof systems is based on the notion of *interactive machines*. An interactive machine is essentially an algorithm that can interact with the external world. An interactive machine can send and receive messages from other interactive machines representing parties in a cryptographic protocol. So this notion makes sense only when considering an interactive machine not individually but in an execution with other interactive machines.

The execution of interactive machines proceeds in *rounds*. In each round, each interactive machine send messages to others and receives messages from

others. Based on the messages received, the interactive machine updates its state and the execution proceeds to the next round. At some point, an interactive machine can *halt* the execution and return some output. In the first rounds, each interactive machine is initialized with some *private input*. We will skip the formalization of interactive machines and protocols for which we defer to the series of books [Foundation of Cryptography, Volume 1, Definition 4.2.1] by Oded Goldreich.

We will focus our study on protocols executed by pairs of interactive machines, that is 2-parties protocols. Given two interactive machines M_0 and M_1, we denote by $\langle \mathsf{M}_0(x_0), \mathsf{M}_1(x_1)\rangle(x)$ the output of M_1 when running on input x_1 and interacting with M_0 running on input x_0 and common input x and by $\mathsf{view}_A\langle A(x_A), B(x_B)\rangle(x)$ (the *view*) all messages received and sent and all random bits (also called *secret coins*) used by A during the interaction with B when both are executed on common input x and A (resp. B) is executed on input x_A (resp. x_B).

Notice that with the notation above we only consider the output of the machine M_1 (and not the output of M_0) because we will be interested in protocols in which machine M_0 is a prover and machine M_1 is a verifier (see later) in which the prover ends with no output and the verifier instead outputs a single bit, i.e., acceptance or rejection. Moreover, observe that in the definition of view_A we just nclude the secret coins that machine A uses in the interaction with B and not also the secret coins that B uses in the interaction with A because we want to define the information that an attacker who would corrupt A can acquire from A.

A *transcript* of the execution of the two machines is the sequence of all messages exchanged by the two machines.

Example of Interactive Proof System. The class of 2-parties protocol we will study is denoted by *interactive proof systems* and we will introduce it with an example. Consider the following scenario. Merlin wants to convince Arthur that he can distinguish between the taste of two quality of beers, Duff and ZKBeer, contained in two unmarked cups. To that purpose, Merlin and Arthur iterate the following process a sufficient number of time, e.g., 30.

Arthur places the two cups behind his back swapping randomly whether Duff will be in the left or right hand. Merlin is not able to see how Arthur swapped the cups behind his back, and in addition Arthur could have asked Merlin to leave the room when performing such a swap. Then, Arthur shows the beers to Merlin and asks him to guess whether Duff is in the left hand or right hand. Merlin tastes both beers and tells Arthur which one contain which quality of beer.

If the two cups contained identical beers (both Duff or both ZKBeer), notwithstanding the tasting ability of Merlin, Merlin will have probability $1/2$ of correctly guessing in each iteration. Instead, if the two cups contain different beers, Merlin will always succed in any experiment using his particular tasting ability.

In the above protocol, Merlin acts like a *prover* interested in proving to

some *verifier* that some statement is correct (in this case, the fact that the two cups contain different qualities of beers). The prover can be dishonest and so the verifier cannot believe the prover on faith. On the other hand, there is an asymmetry of computational power: the verifier needs the prover to be sure about the validity of the statement because the verifier is not able to verify the statement by itself (in this case, Arthur has no tasting ability). Two properties must hold. If the statement is true, the prover should always succeed in convincing the verifier; this property is called the *completeness*. If the statement is false, no malicious prover should be able to convince the verifier, except with small probability; this property is called *soundness*. We will focus our attention on IPS for polynomial-time relations in which the prover has to prove that some input x belongs to some NP language having in addition the witness to the fact that $x \in L$. The verifier will be executed on the common input x but w is known only by the prover. We formalize this as follows.

Definition 42 (Interactive proof system) *A pair $(\mathcal{P}, \mathcal{V})$ of PPT interactive machines is a* interactive proof system *(IPS) for polynomial-time relation $\mathcal{R}$ associated with a language L if the following properties of completeness and soundness hold:*

- Completeness. *For every $(x, w) \in \mathcal{R}$, it holds that:*

$$\text{Prob}[\langle \mathcal{P}(w), \mathcal{V} \rangle(x) = 1] = 1.$$

- Statistical soundness. *For every non-uniform (possibly computationally unbounded) machine $\mathcal{P}^\star \stackrel{\triangle}{=} \{\mathcal{P}^\star_\lambda\}_\lambda$, it holds that for every polynomial $p(\cdot)$, there exists a constant n such that for every $\lambda \geq n$, for every $x \notin L, x \in \{0,1\}^{\leq \lambda}$, it holds that:*

$$\text{Prob}[\langle \mathcal{P}^\star_\lambda, \mathcal{V} \rangle(x) = 1] \leq 1/p(\lambda).$$

  The statistical soundness can be generalized to $s(\cdot)$-soundness as follows.

  Definition 43 ($s(\cdot)$-soundness) *Let $s(\cdot)$ be a function. An interactive proof system $(\mathcal{P}, \mathcal{V})$ for polynomial-time relation $\mathcal{R}$ associated with a language L satisfies $s(\cdot)$-soundness if the following holds. For every non-uniform (possibly computationally unbounded) algorithm $\mathcal{P}^\star \stackrel{\triangle}{=} \{\mathcal{P}^\star_\lambda\}_{\lambda>0}$, it holds that there exists a constant n such that for every $\lambda \geq n$, for every $x \notin L \cap \{0,1\}^\lambda$, it holds that:*

$$\text{Prob}[\langle \mathcal{P}^\star_\lambda, \mathcal{V} \rangle(x) = 1] \leq 1/s(\lambda).$$

A particular case of IPS is the one of public-coin protocols, that is protocols in which the messages sent by verifier to the prover consist of just random strings.

Public here means that the messages sent by the verifier to the prover (that consist of random strings) can be read by an eavesdropper. We remark that the public-coin property implies that, in any round, the messages of the verifier are completely independent from the past messages sent by the prover and from the inputs of the two parties.

Definition 44 (Public-coin protocol) *An interactive proof system $(\mathcal{P}, \mathcal{V})$ is public-coin if the messages sent by $\mathcal{V}$ during the execution of the protocol (on any input) are random and independently chosen coins. Precisely, there exists a sequence of families of finite sets of binary strings $\{D_{\lambda,1}\}_{\lambda>0}, \ldots$ of length polynomial in λ such that for any $x \in L$ and for each $i > 0$, $\mathcal{V}$ sends a string sampled by $U_{|D_{|x|,i}|}$ as its i-th message of the protocol on common input x.*

Definition 45 (Perfect soundness) *An interactive proof system $(\mathcal{P}, \mathcal{V})$ for polynomial-time relation $\mathcal{R}$ associated with a language L satisfies perfect soundness (or it is perfectly sound) if it satisfies* 0*-soundness.*

We now formalize the notion of zero-knowledge for IPS. Intuitively, am IPS is zero-knowledge if the verifier, at the end of the execution of the protocol, gains no additional information that the verifier could not have computed by itself before executing the protocol. This intuition is formalized stating the existence of an efficient *simulator* algorithm that can generate the view of the verifier in the protocol without using any secret input owned by the prover. That is, the verifier could have used the simulator to generate the view and so this view does not give the verifier additional information. We will focus our attention on protocols in which the verifier acts semi-honestly, that is in which the verifier honestly follows the protocol but is interested in leaking additional information about the witness of the prover.

Definition 46 (Computational and statistical honest verifier zero-knowledge) *A proof system for a polynomial-time relation $\mathcal{R}$ consisting of a pair $(\mathcal{P}, \mathcal{V})$ of PPT interactive machines is called a computational (resp. statistical)* honest verifier zero-knowledge *(HVZK, in short) if it satisfies the following computational (resp. statistical) honest verifier zero-knowledge property.*

- Computational Honest Verifier Zero-Knowledge: *There exists a PPT algorithm* Sim *(called the* simulator *for $\mathcal{V}$) such that for every sequence $\{(x_\lambda, w_\lambda)\}_{\lambda>0}$ such that for every $\lambda > 0$, $(x_\lambda, w_\lambda) \in \mathcal{R}$, for every polynomial $p(\cdot)$, there exists a number $n > 0$ such that for every $\lambda \geq n$, no non-uniform PPT distinguisher algorithm can distinguish the following two sequences of random variables with advantage $> 1/p(\lambda)$:*
  - $\{\mathsf{view}_{\mathcal{V}}\langle \mathcal{P}(w_\lambda, U_m), \mathcal{V}\rangle(x_\lambda)\}_{\lambda \geq n}$. *(Where m is the number of random coins $\mathcal{P}$ uses). (Cf. Def. 14 for the notation $\mathcal{P}(w_\lambda, U_m)$.)*
  - $\{\mathsf{Sim}(x_\lambda)\}_{\lambda \geq n}$.

Statistical Honest Verifier Zero-Knowledge: *This is identical to computational honest verifier zero-knowledge except that it is quantified for every non-uniform (possibly computationally unbounded) distinguisher algorithms.*

The issue of interaction for verifiability. Recall that our goal was to use ZK proofs to add verifiability to the first iteration of our toy e-voting scheme.

The issue in using interactive proof systems is that the transcript of a ZK interactive proof is *simulatable* and thus would only convince the verifier who actually interacted and participated in procuding such a transcript but not external parties who did not participate in the execution. Therefore, interactive ZK proofs are not suitable to achieve *universal verifiability*, that is the property that any party, not just who participated in the execution, should be able to verify the proofs. To this purpose, we need non-interactive ZK proofs. Unfortunately, they provably do not exist. We will first demonstrate this fact and then in Section 3.3 we will show how non-interactive ZK is instead achievable in a special model of practical interest.

Definition 47 (Non-interactive proof system in the plain model) *A pair of PPT (non-interactive) algorithms $(\mathcal{P}, \mathcal{V})$ is a perfectly sound (or statistically sound) non-interactive proof (NI) system (in the* plain model*)*[9] *for polynomial-time relation $\mathcal{R}$ associated with a language L if $(\mathcal{P}, \mathcal{V})$ satisfy the following properties:*

-
- Completeness. *For every $(x, w) \in \mathcal{R}$, it holds that:*

$$\text{Prob}[\mathcal{V}(x, \mathcal{P}(x, w)) = 1] = 1.$$

- Statistical soundness. *For every polynomial p, there exists a constant $n_0 \in \mathbb{N}$ such that for every $\lambda > n$, for every $x \notin L, x \in \{0,1\}^{\leq \lambda}$, for every string $\pi \in \{0, 1, \star\}$ it holds that:*

$$\text{Prob}[\mathcal{V}(x, \pi) = 1] \leq 1/p(\lambda).$$

- Perfect soundness *For every $x \notin L$, for every $\pi \in \{0, 1, \star\}$, $\mathcal{V}(x, \pi) = 0$.*

For any $(x, w) \in \mathcal{R}$, we call any string π in the range of $\mathcal{P}(x, w)$ a proof for x.

Note that the main difference is that $\mathcal{P}$ and $\mathcal{V}$ are PPT algorithms and not interactive algorithms.

Definition 48 (NIZK) *A non-interactive ZK system for a polynomial-time relation $\mathcal{R}$ associated with a language L is a NI $(\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ if there exists a PPT algorithm (called the simulator) that satisfies the following (perfect, statistical, or computational) zero-knowledge (ZK) properties:*

- Perfect ZK. *For any pair $(x, w) \in \mathcal{R}$, the random variable $\mathsf{Sim}(x; U_m)$ is distributed identically to the output of $\mathcal{P}(x, w; U_m)$. (Where m is the maximum number of random coins (that can be a polynomial function of the length of x) that $\mathcal{P}$ and Sim use).*

[9] Here, by plain model we mean that the system is not based on trust assumption like in the *Common Reference String* model or in the Random Oracle model.

- Statistical ZK. *There exists a negligible function* negl *such that there exists* $n_0 \in N$ *such that for any pair* $(x, w) \in \mathcal{R}$ *with* $|x| > n_0$*, the random variable* $\mathsf{Sim}(x)$ *is* $\mathsf{negl}(|x|)$*-close to* $\mathcal{P}(x, w)$*.*

- Computational ZK. *For every sequence* $\{(x_\lambda, w_\lambda)\}_{\lambda>0}$ *such that for every* $\lambda > 0$*,* $(x_\lambda, w_\lambda) \in \mathcal{R}$*, for every polynomial* $p(\cdot)$*, there exists a number* $n > 0$ *such that for every* $\lambda \geq n$*, no non-uniform PPT distinguisher algorithm can distinguish the following two sequences of random variables with advantage* $> 1/p(\lambda)$*:*

  - $\{\mathcal{P}(x_\lambda, w_\lambda; U_m)\}_{\lambda \geq n}$.
  - $\{\mathsf{Sim}(x_\lambda; U_m)\}_{\lambda \geq n}$.

  (Where $m(\cdot)$ *is the maximum number of random coins (that can be a polynomial function of length of* x*) that* $\mathcal{P}$ *and* Sim *use.)*

Note that in the above definition we require $(\mathcal{P}, \mathcal{V})$ to be non-interactive algorithms and thus the prover, on input (x, w) (and the random coins), outputs a proof π and the verifier $\mathcal{V}$, on input two strings x and π, outputs its decision on whether $x \in L$. Observe that statistical soundness in case of PPT algorithms accounts to rejecting proofs for false statements with negligible probability in the random coins used by the verifier only.

Theorem 10 (impossibility of NIZK in the plain model) *Let* L *be a hard language. There is no NIZK in the plain model for* L*.*

Suppose towards a contradiction that $(\mathcal{P}, \mathcal{V})$ is a pair of PPT algorithms satisfying (perfect or statistical) soundness and (perfect, statistical or computational) ZK. We construct a PPT algorithm A that can decide membership in L contradicting the hardness of L. By ZK, there exists a PPT algorithm Sim that satisfies (perfect, statistical or computational) ZK. The algorithm A works as follows: A takes as input a string x, compute $\pi \leftarrow \mathsf{Sim}(x)$ and outputs $\mathcal{V}(\pi)$ as its decision on accepting or rejecting x. By the def. of statistical soundness, if $x \notin L$, there exists a polynomial p and a constant $n_0 > 0$ such that for every integer $\lambda \geq n_0$, for every string $\pi \in \{0, 1, \star\}$, $\mathcal{V}(x, \pi) = 0$ except with prob. $\leq 1/p(|x|)$. Therefore, by construction of A, if $x \notin L$, there exists a polynomial p and a constant $n_0 > 0$ such that for every integer $\lambda \geq n_0$, for every string $\pi \in \{0, 1, \star\}$, $\mathcal{V}(x, \pi) = 0$ except with prob. $\leq 1/p(|x|)$.

Suppose that $x \in L$. Perfect ZK implies that each string π output by the simulator has equal probability of occurring as a string output by $\mathcal{P}(x, w)$ where $(x, w) \in \mathcal{R}$. By completeness $\mathcal{V}(x, \pi) = 1$. Therefore, perfect ZK implies that for every $x \in L, \mathcal{V}(x, \mathsf{Sim}(x)) = 1$ and thus $A(x) = 1$. We leave as exercise to verify that, by Fact 5, this also holds for statistical and computational ZK. So A decides L, a contradiction.

3.2 Sigma Protocols

We will now present a class of IPS of practical interest that we will later employ to construct our toy e-voting scheme.

Definition 49 (Σ-protocol) *An interactive protocol* $(\mathcal{P}, \mathcal{V})$ *is a* Σ-protocol *for a polynomial-time relation* $\mathcal{R}$ *associated with a language* L *if it enjoys the following properties:*

- Public-coin 3-move. $(\mathcal{P}, \mathcal{V})$ *is a 3-move public-coin protocol. We denote the transcript of an execution of a* Σ*-protocol by a triple of messages* (a, c, z)*, where* a *and* z *are sent by* $\mathcal{P}$ *and* c*, the* challenge*, is* $\mathcal{V}$*'s only message that is, by the public-coin property, a random string. We say that a transcript* (a, c, z) *is* accepting *(or valid) if* $\mathcal{V}(a, c, z)$ *outputs 1.*

  Since the protocol is public-coin, there exists a family of finite sets $\{D_\lambda\}_{\lambda>0}$ *of length polynomial in* λ *such that, for any input* x *and first message* a*,* $\mathcal{V}$ *sends a random string* $c \leftarrow D_\lambda$*. We call* D_λ *the* challenge domain *of the protocol.*

- Completeness. *For every* $(x, w) \in \mathcal{R}$*, it holds that*

$$\text{Prob}[\langle \mathcal{P}(w), \mathcal{V} \rangle(x) = 1] = 1.$$

- Special Soundness. *There exists a PPT algorithm* Extract *that, on input* x *and any pair of accepting transcripts for* x*,* $(a, c, z), (a, c', z')$*, where* $c \neq c'$*, outputs* w *such that* $(x, w) \in \mathcal{R}$*.*

- Honest Verifier ZK (HVZK). *There exists a PPT* simulator *algorithm* Sim *that, on input an instance* $x \in L$*, outputs* (a, c, z) *such that* (a, c, z) *has the same distribution of transcripts obtained when* $\mathcal{V}$ *sends* c *as challenge and* $\mathcal{P}$ *runs on common input* x *and any private input* w *such that* $(x, w) \in \mathcal{R}$*.*

A Σ*-protocol satisfies special honest verifier ZK if the following holds:*

- Special Honest Verifier Zero Knowledge (SHVZK). *There exists a PPT* simulator *algorithm* Sim *that, on input an instance* $x \in L$ *and a challenge* c*, outputs* (a, c, z) *such that* (a, c, z) *has the same distribution of transcripts obtained when* $\mathcal{V}$ *and* $\mathcal{P}$ *runs on common input* x *and any private input* w *such that* $(x, w) \in \mathcal{R}$*.*

Special soundness defined above generalizes soundness in the following sense.

Theorem 11 (relation between special soundness and soundness) *Let* $\mathcal{R}$ *be a polynomial-time relation associated with a language* L*. A* Σ*-protocol* $(\mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *and challenge domain* D_λ *satisfies* $\frac{1}{|D_\lambda|}$*-soundness.*

Proof 11 *Let* $x \notin L$*. Suppose towards a contradiction that there exist two accepting transcripts* (a, c, z) *and* (a, c', z) *for* x*. Then, by special soundness* Extract*, on input such two transcripts, computes a string* w *such that* $(x, w) \in \mathcal{R}$ *contradicting the fact that* $x \notin L$*. Therefore, for any* a *there exists at most a single challenge* c *such that there exists a* z *such that* (a, c, z) *is an accepting transcript for* x*. Since* c *is a randomly chosen string in* $D_{|x|}$*, it follows that, except with probability* $\frac{1}{|D_{|x|}|}$*,* $\mathcal{V}$ *rejects* x*.*

Henceforth, we assume $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda>0}$ to be a family of groups of prime order q_λ that are in the range of some generator Gen of families of groups.

Definition 50 ($\mathcal{R}_{\mathsf{DDH}}$) *The relation of well-formedness of Diffie-Hellman (DH) tuples (for the family of groups $\mathcal{G}$) is defined as follows: $\mathcal{R}_{\mathsf{DDH}}((\mathbb{G}, q, g, h, u, v), w) = 1$ iff $\mathbb{G} \in \mathcal{G}$, $g, h, u, v \in \mathbb{G}$ and $u = g^w, v = h^w$ for some non-negative integer $w < \mathbb{Z}_q$. It is easy to see that this is a polynomial-time relation.*

We now show a Σ-protocol for $\mathcal{R}_{\mathsf{DDH}}$.

Definition 51 (Σ_{DDH}) Σ_{DDH} *consists of the following pair of interactive machine $(\mathcal{P}, \mathcal{V})$. Let L be the language associated with $\mathcal{R}_{\mathsf{DDH}}$.*

- *Inputs. $\mathcal{P}$ and $\mathcal{V}$ start with the common input $x \stackrel{\triangle}{=} (\mathbb{G}, q, g, h, u, v) \in L$ and $\mathcal{P}$ additionally takes as input the witness $w \in \mathbb{Z}_q$.*
- *First message. $\mathcal{P}$ chooses a random $r \in \mathbb{Z}_q$ and sends $(a \stackrel{\triangle}{=} g^r, b \stackrel{\triangle}{=} h^r)$ as first message.*
- *Challenge. $\mathcal{V}$ sends a random element $c \leftarrow \mathbb{Z}_q$ as its challenge.*
- *Response. $\mathcal{P}$ responds to the challenge sending $z \stackrel{\triangle}{=} r + c \cdot w \mod q$.*
- *Verifier's decision. $\mathcal{V}$ on input x, the first message (a, b), the challenge c and the response z outputs 1 iff*

$$g^z = a \cdot u^c, h^z = b \cdot v^c.$$

Theorem 12 *Σ_{DDH} is a Σ-protocol for $\mathcal{R}_{\mathsf{DDH}}$.*

Proof 12 *We show that completeness, special soundness, and HVZK hold.*

- *Completeness. It is straightforward to see that completeness holds.*
- *Special soundness. Given (a, b, c, z), (a, b, c', z'), we have $g^z = a \cdot u^c, g^{z'} = au^{c'}, h^z = b \cdot v^c, h^{z'} = b \cdot v^{c'}$ and so (can be seen that) $w = (z - z')/(c - c')$. So the algorithm that on input two transcripts (a, b, c, z), (a, b, c', z') for the same first message, outputs $(z - z')/(c - c')$ is able to efficiently compute a witness, as we had to prove.*
- *HVZK. Consider the following simulator algorithm Sim. Given $x \stackrel{\triangle}{=} (\mathbb{G}, q, g, h, u, v)$, Sim chooses random $c, z \leftarrow \mathbb{Z}_q$ and outputs*

$$a = q^z \cdot u^{-c}, b = h^z \cdot v^{-c}.$$

It is straightforward to check that the random variable of the output by Sim on input $x \in L$ is identically distributed to the random variable of $\mathcal{P}(x, w)$ for any $(x, w) \in \mathcal{R}_{\mathsf{DDH}}$.

3.2.1 OR composition of Σ-protocols

For our e-voting scheme, it will be particularly useful to consider the OR composition of Σ-protocols as defined next. That is, we will need to consider statements of the form: either $x_1 \in L$ or $x_2 \in L$ with respect to an NP-language L. The prover will have the witness for either of the two statements but the verifier should not leak which particular statement among the two is true.

Definition 52 (OR composition of Σ-protocols) *Let $\Sigma \stackrel{\triangle}{=} (\mathcal{P}', \mathcal{V}')$ be a Σ-protocol satisfying SHVZ for polynomial-time relation $\mathcal{R}$ with challenge domain $\{\{0,1\}^\lambda\}_{\lambda>0}$ (i.e., the verifier, on an input of length λ, will send an uniform string in $\{0,1\}^\lambda$ as their challenge). Consider the following relation: $\mathcal{R}_{\mathsf{OR},\mathcal{R}}((x_1,x_2),w) = 1$ iff $\mathcal{R}(x_1,w) = 1 \ \vee \ \mathcal{R}(x_2,w) = 1$. For simplicity, sometimes we will omit the subscript $\mathcal{R}$.*

We construct the following protocol $\Sigma_{OR} = (\mathcal{P}, \mathcal{V})$.

- *Inputs. Let L be the language associated with $\mathcal{R}_{\mathsf{OR}}$. $\mathcal{P}$ and $\mathcal{V}$ start with the common input $x \stackrel{\triangle}{=} (x_1,x_2) \in L$ and $\mathcal{P}$ additionally takes as input the witness w such that $(x,w) \in \mathcal{R}_{\mathsf{OR}}$. By the definition of $\mathcal{R}_{\mathsf{OR}}$ this means that either $(x_1,w) \in \mathcal{R}$ or $(x_2,w) \in \mathcal{R}$. Suppose w.l.o.g that $(x_1,w) \in \mathcal{R}$.*
- *First message. Let Sim' be the PPT simulator associated with Σ (guaranteed by the SHVZK property). $\mathcal{P}$ chooses $c_2 \leftarrow \{0,1\}^{|x_2|}$. and runs $\mathsf{Sim}'(x_2,c_2)$ to get (a_2,z_2). $\mathcal{P}$ runs $\mathcal{P}'(x_1,w_1)$ to get a_1. $\mathcal{P}$ sends (a_1,a_2) as its first message.*
- *Challenge. $\mathcal{V}$ replies with $c \leftarrow \{0,1\}^{|x_1|}$.*
- *Response. $\mathcal{P}$ computes $c_1 = c \ \oplus \ c_2$ and computes the response z_1 of $\mathcal{P}'$ to c_1. $\mathcal{P}$ sends $Z \stackrel{\triangle}{=} (c_1,c_2,z_1,z_2)$ as its response.*
- *Verifier's decision. $\mathcal{V}$ accepts iif (1) $c = c_1 \ \oplus \ c_2$ and both (2) (a_1,c_1,z_1) and (3) (a_2,c_2,z_2) are accepting transcripts resp. for $\mathcal{V}'(x_1,(a_1,c_1,z_1)$ and $\mathcal{V}'(x_2,(a_2,c_2,z_2)$.*

Theorem 13 *Let $\mathcal{R}$ be a polyonomial-time relation with challenge domain $\{\{0,1\}^\lambda\}_{\lambda>0}$. Σ_{OR} is Σ-protocol for $\mathcal{R}_{\mathsf{OR},\mathcal{R}}$ with associated language L.*

Proof 13 *We prove that completeness, special soundness and SHVZK hold.*

- *Completeness. It is easy to see that completeness holds.*
- *Special soundness. Let $(a \stackrel{\triangle}{=} (a_1,a_2), c, Z_1 \stackrel{\triangle}{=} (c_1,c_2,z_1,z_2), (a \stackrel{\triangle}{=} (a_1,a_2), c, Z_2 \stackrel{\triangle}{=} (c_1',c_2',z_1',z_2'))$ be two accepting transcripts for an input $x \in L$. By $\mathcal{V}$'s check (1), $c_1 \neq c_1'$ and $c_2' \neq c_2'$ (in fact, if $c_1 = c_1'$ and $c_2 = c_2'$ then $c = c'$, contradicting the hypothesis). By $\mathcal{V}$'s checks (2) and (3), the two transcripts (a_1,c_1,z_1) and (a_1,c_2,z_2) are accepting for x_1, so the extractor of σ guaranteed by special soundness can efficiently extract w, as we had to prove*

- *SHVZK. We construct the following simulator* Sim *using the simulator* Sim' *guaranteed by the SHVZK property of* σ. Sim *takes as input* $x \stackrel{\triangle}{=} (x_1, x_2) \in L$. *Given a challenge* $c \in \{0,1\}^{|x_1|}$, *it chooses random* c_1, c_2 *s.t.* $c_1 \oplus c_2 = c$ *and for* $i = 1, 2$ *computes* $(a_i, z_i) \leftarrow \mathsf{Sim}'(x_i, c_i)$ *and outputs* $((a_1, a_2), c, (c_1, c_2, z_1, z_2)$ *as its output. By the SHVZK property of* σ *and by construction of* Σ_{OR}, *it is easy to see that the output of* Sim *has the same distribution as* $\mathcal{P}(x, w)$ *where* $(x, w)\mathcal{R}_{\mathsf{OR}}$.

3.3 The Random Oracle Model and the Fiat-Shamir Transform

We saw that NIZK proofs in the plain model do not exist. Fortunately, there exists an alternative model in which NIZK proofs can be implemented in practice: the Random Oracle (RO) model. First of all, we need to introduce the notion of a (cryptographic) hash function.

Definition 53 (sketch) *An hash function* H *is an efficient algorithm that maps an arbitrary string* $x \in \{0,1\}^\star$ *to a short output* $y \in \{0,1\}^m$ *for some integer* $m > 0$. *For instance, the famous SHA256 hash functions maps strings of arbitrary length to strings of 256 bits. An hash function should at least guarantee two basic security properties:*

- *Preimage resistance. There is no PPT adversary* A *such that for any* $n > 0$, $\text{Prob}[A(y) = H^{-1}(y) | x \leftarrow \{0,1\}^n,\ y = H(x)]$ *is negligible in* n, *that is* A *can find a preimage to* y *only with negligible probability.*
- *Collision resistance. There is no PPT adversary* A *that can output two inputs* x_1 *and* x_2 *such that* $H(x_1) = H(x_2)$. *In this case,* x_1 *and* x_2 *are called* collisions *of* H.

The previous definition is actually impossible to achieve. The reason is that collisions of H always exist, and then an algorithm could have a collision embedded into its code and just output it. To formalize hash functions, one would need the notion of *keyed* hash functions for which we defer to [Katz and Lindell, An introduction to Modern Cryptography, Chapter 4]. In practice, however, hash functions like SHA256 can be thought as being collision resistance since no collisions are currently known for it.

Exercise 10 *The SHA256 hash function* H *takes as input strings of arbitrary length and maps it into strings of* 256 *bits. Show how to use Thm. 1 to find collisions in SHA256 in about* 2^{128} *steps.*

In the RO methodology, we assume that all parties of a protocol have access to an ideal random function O. This means that all parties can query the function O on an input x and get as answer an output $y \stackrel{\triangle}{=} O(x)$ that is uniformly chosen in some domain. If two parties query the oracle on the same input, they get the same answer. The random oracle might be implemented as a

random table but the size of the table would be too huge to be stored. In practice, the random oracle is replaced by a (cryptographic) hash function. The security of the protocol is analyzed assuming that the parties have access to the ideal random function but then the RO methodology states that replacing the RO with a concrete hash function the protocl retains its security. There are contrived examples in which the RO methodology fails. However, no practical real-world protocol based on the RO methodology has been broken.

The Fiat-Shamir Transform. In the RO model it is possible to remove interaction from interactive ZK proof systems using the so called *Fiat-Shamir (FS) transform*.

Let $\Sigma \triangleq (\mathcal{P}, \mathcal{V})$ be a Σ-protocol for some polynomial-time relation relative to a language L. The FS transform constructs $\mathsf{NIZK} = (\mathsf{NIZK}.\mathcal{P}, \mathsf{NIZK}.\mathcal{V})$ as follows. $\mathsf{NIZK}.\mathcal{P}$ computes a value a precisely as $\mathcal{P}$ would, but then the challenge c of $\mathcal{V}$ is replaced by the output of the RO on input the statement x and a, i.e., $c = H(x, a)$. Finally $\mathsf{NIZK}.\mathcal{P}$ computes z precisely as $\mathcal{P}$ would.

NIZK is only sound against nuPPT adversaries (i.e., it is an *argument* system)[10] in the random oracle model; this suffices in applications.

The ZK property can be proven assuming that the simulator to be able to *program* the RO to its liking. Essentially, the simulator would compute a transcript (a, c, z) as a simulator of Σ would do and then would program the RO so that $H(x, a) = c$. Observe that in this model we can bypass the impossibility result of NIZK in the plain model because such a simulator cannot be used to decide L. We skip the details for which we defer to `https://eprint.iacr.org/2012/704`.

As by product, we can turn all Σ-protocols of Section 3.2 into NIZK

Exercise 11 (FS transform for general public-coin protocols) *Show that the FS transform can be also applied to general interactive public-coin protocol and not just Σ-protocols.*

3.4 Verifiable e-voting from NIZK proofs

Our toy e-voting scheme: second iteration. We saw that our first iteration of our toy e-voting scheme suffers the problem of verifiability. We now demonstrate how NIZK proofs derived from Σ-protocols via the FS transform can be used to add universal verifiability to our e-voting scheme. Recall that we need to add two types of proofs.

- Proofs of correct decryption. The authority A homomorphically tallies all ElGamal ciphertext to get a ciphertext $\mathsf{CT} \triangleq (\mathsf{CT}_1, \mathsf{CT}_2)$ encrypting the tally v of the election and has to prove that CT actually decrypts to v (and not to any other result).

[10] Some authors are pedant in using the terminology of argument system for systems enjoying only computational soundness. We will instead simplify using the term NIZK proof also for NIZK arguments.

We can solve this problem in the following way. Let Σ_{DDH} be a Σ-protocol for $\mathcal{R}_{\mathsf{DDH}}$ and let $\mathsf{NIZK} = (\mathsf{NIZK}.\mathcal{P}, \mathsf{NIZK}.\mathcal{V})$ be a NIZK for the same relation derived from Σ_{DDH} via the FS transform. Recall that $\mathsf{CT} = (g^r, h^r \cdot g^v)$ for some integer $r \in \mathbb{Z}_q$, where $h = g^w$ is the PK for SK $w \in \mathbb{Z}_q$. Observe that $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_1^w \cdot g^v$. Moreover, notice that $(g, \mathsf{CT}_1, h, \mathsf{CT}_2 \cdot g^{-v}) = (g, \mathsf{CT}_1, g^w, \mathsf{CT}_1^w)$, so it is a DH tuple. The viceversa also holds: if $(g, \mathsf{CT}_1, h, \mathsf{CT}_2 \cdot g^{-v})$ is a DH tuple then $\mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_1^w \cdot g^v$.) and so CT decrypts to v.

So the authority A can use NIZK to provide a (non-interactive) proof of the fact that CT decrypts to v as we wanted.

- Proofs of correct encryption. The voter needs to prove that his ciphertext $\mathsf{CT} \stackrel{\triangle}{=} (\mathsf{CT}_1, \mathsf{CT}_2)$ transmitted to the PBB encrypts one of the two valid options, 0 or 1. We achieve this in the following way.

  Let Σ_{OR} be Σ-protocol for $\mathcal{R}_{\mathsf{OR},\mathcal{R}_{\mathsf{DDH}}}$ and let L be the associated language and let $\mathsf{NIZK} = (\mathsf{NIZK}.\mathcal{P}, \mathsf{NIZK}.\mathcal{V})$ be a NIZK for the same relation derived from Σ_{OR} via the FS transform.

  Recall that $\mathsf{CT} = (g^r, h^r \cdot g^v)$ for some integer $r \in \mathbb{Z}_q$, where h is the PK.

  Then let $\mathsf{CT}' \stackrel{\triangle}{=} (\mathsf{CT}_1, \mathsf{CT}_2 \cdot g^{-1}$. Notice that $v \in \{0, 1\}$ if and only if either CT is encryption of 0 or CT' is encryption of 0. Moreover, observe that CT is encryption of 0 iff $(g, h, \mathsf{CT}_1, \mathsf{CT}_2)$ is a DDH tuple and CT' is encryption of 1 iff iff $(g, h, \mathsf{CT}_1, \mathsf{CT}_2 \cdot g^{-1})$ is a DDH tuple. So, the voter can use NIZK to prove that either CT or CT' is a DDH tuple and this would be a proof that $v \in \{0, 1\}$.

Hence, in this second iteration we added universal verifiability to our e-voting scheme. The last issue left is the one of privacy. The single authority A has the possibility of decrypting individual ciphertexts of each voter (not just the ciphertext resulting from the homomorphic tallying) completely breaking the privacy. In Section 4 we will address this issue.

4 Secret Sharing and Threshold Cryptography

A (t, n)-secret sharing scheme (SS) is a mechanism for n parties to share a secret s among parts $s_1, \ldots, s_n$ such that a subset of cardinality $t + 1$ of these parts allows to reconstruct the secret. The secret sharing scheme is said to be *perfect* if no subset of less than $t + 1$ shares leaks any information regarding the secret, that is, given any subset of $n - 1$ shares, any other value $s' \neq s$ has the same probability of being the secret.

A $(n - 1, n)$-SS can be implemented as follows. Generate the first $n - 1$ shares $s_1, \ldots, s_{n-1}$ at random and set $s_n = s - \sum_{i=1}^{i=n-1} s_i$. The secret s can be recovered as $s = \sum_{i=1}^{n} s_i$, but it is straightforward to see that no subset of $n-1$ parties can reconstruct the secret s, that is such SS scheme is perfect.

Shamir's (t, n)-threshold scheme. Shamir's scheme is based on Lagrange's polynomial interpolation that we recall.

Theorem 14 (Lagrange's interpolation) *Given $t+1$ distinct points (x_i, y_i) of the form $(x_i, p(x_i))$, where $p(x)$ is a polynomial of degree t, and given a subset Q of $[n]$ of cardinality $t+1$, then $p(x)$ is determined by*

$$p(x) = \sum_{i=1}^{t+1} y_i \cdot \prod_{j \in Q, j \neq i} \frac{x - x_j}{x_i - x_j}.$$

Shamir's (t, n)-SS is defined over the field $\mathbb{Z}_p^\star$, for prime p. There is a *trusted dealer* who chooses a random secret $s \leftarrow Z_p^\star$, sets $a_0 = s$ and chooses $a_1, \ldots, a_t$ at random in $\mathbb{Z}_p^\star$.

The trusted party dealer computes $y_i = p(i)$ for any $i \in [n]$, where $p(x) \stackrel{\triangle}{=} \sum_{i=0}^{t} a_i \cdot x^i$, and distributes share y_i to party i. The secret is the constant term $a_0 = s$ and the secret can be recovered by any subset Q of cardinality $t+1$ by the formula:

$$p(0) = \sum_{i=1}^{t+1} \lambda_i \cdot y_i,$$

where

$$\lambda_i = \prod_{j \in Q, j \neq i} \frac{j}{j - i}.$$

It is easy to see that the above scheme is a $(t.n)$-SS and is perfect.

4.1 Applications of Secret Sharing to e-voting

Recall that in the ElGamal PKE scheme the PK has the form $h = g^w$, where w is the corresponding SK. In the current iteration of our toy e-voting scheme there is a single authority A who owns this SK and so can decrypt every single ciphertext and break the privacy of the voters. We now show how to reduce the trust in a single authority using a $(n-1, n)$-SS scheme.

Third iteration of our toy e-voting scheme. Instead of a single authority, we suppose there are n authorities $A_1, \ldots, A_m$. For any $i \in [m]$, A_i setups a pair of ElGamal public- and secret- keys $(\mathsf{PK}_i = g^{w_i}, \mathsf{SK}_i = w_i)$. The setup is done running the setup of the ElGamal PKE scheme without any modification. Each authority A_i publishes on the PBB its own PK PK_i and from the m values PK_i, everyone (not just the authorities) can compute the *general public-key* PK defined so that: $\mathsf{PK} \stackrel{\triangle}{=} \prod_{i \in [m]} \mathsf{PK}_i = \prod_{i \in [m]} g^{w_i} = g^{\sum_{i \in [m]} w_i}$.

We will so denote by $w \stackrel{\triangle}{=} \sum_{i \in [m]} w_i$ the *general secret-key*. Notice that the general SK is implicitly generated by the n authorities' secret-keys w_i's but, by the properties of the SS, no subset of $n-1$ authorities can reconstruct the general SK.

Given the general PK, any voter can proceed as in the previous iteration of our toy e-voting scheme: they encrypt and they provide proofs of correct encryption with respect to the general PK. In fact, from the point of view of the voters, there is no change.

We now show how the m authorities decrypt the tally and provide joint proofs of correct decryption. As before, given n ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ encrypting the prefences $v_1, \ldots, v_n$ of the n voters, everyone can use the homomorphic properties of ElGamal to produce a ciphertext CT encrypting $M \triangleq g^{\sum_{i\in[n]} v_i}$.

Observe that CT has the form $(g^r, \mathsf{PK}^r) = (g^r, M \cdot g^{r\cdot\sum_{i\in[m]} w_i})$. For any $i \in [m]$, let $\mathsf{CT}^{(i)} \triangleq (g^r, M \cdot g^{r\sum j\in[m],j>i w_j} \cdot g^{rw_i}$. We have that: $\mathsf{CT} = \mathsf{CT}^{(1)}$, and $\mathsf{CT}^{(m)}$ encrypts M.

At this point each authority uses its SK w_i to decrypt $\mathsf{CT}^{(i)}$, and it is easy to see that the result of the decrpytion of $\mathsf{CT}^{(i)}$ is a ciphertext $(\mathsf{CT}', \mathsf{CT}'')$ such that $\mathsf{CT}' = g^r$ and $\mathsf{CT}'' = M \cdot g^{r\sum j\in[m],j>i w_i}) = M \cdot g^{r\sum j\in[m],j>i+1 w_j} \cdot g^{rw_{i+1}}$. So, after decrypting $\mathsf{CT}^{(i)}$, we have the ciphertext $\mathsf{CT}^{(i+1)}$.

Furthermore, each ciphertext $\mathsf{CT}^{(i)}$ is a valid ElGamal ciphertext with respect to PK g^{w_i} and message $M \cdot g^{r\sum j\in[m],j>i w_j}$, so the proof of correct decryption of Section 3.4.

The m-th authority A_m will so decrypt the result $M \triangleq g^{\sum_{i\in[n]} v_i}$ along with proof of correct decryption and announce the result $v \triangleq \sum_{i\in[n]} v_i$. All intermediate decrypted values along with the corresponding proofs of correct decryption (posted by each authority) are posted on the PBB and all together prove that v is the correct result encrypted in CT.

We showed how to reduce the trust for privacy from a single authority to m ones. The problem in this solution is in the *robustness,* that is if only one of the authorities has a fault or stop collaborating, the result of the election cannot be computed. Shamir's (t, m)-SS could be used to improve robustness in a slightly more complex way but we omit details.

We now highlight other unsolved issues in the current iteration of our toy e-voting scheme. Every party using the PK published on the PBB can send to the PBB an encrypted vote. So, a malicious user could send multiple votes. To withstand this attack we need to bind the legitimate voters to some digital identity so that only ciphertexts carrying some valid *digital signature* of a legitimate voter have to be counted.

Moreover, observe that the ElGamal ciphertexts are re-randomizable in the sense that given a ciphertext CT encrypting an unknown integer v it is possible to compute a different ciphertext CT' encrypting v for some v'. An adversary could intercept a ciphertext directed to the PBB, re-randomize it, and send it to the PBB resulting into a copy of the vote encrypted in CT. The adversary would not know the vote he is implicitly submitting but he would be able to submit the same vote and this may be harmful in some settings.

These attacks can be defeated using digital signatures that will be presented in Section 5.1.

4.2 Secure Function Evaluation from Secret Sharing

In this Section, we show how SS can be used to solve a fundamental task in cryptography, namely the problem of *secure function evaluation (SFE).* We have a set of parties $P_1, \ldots, P_n$ holding resp. secret inputs $x_1, \ldots, x_n$ and they want to jointly compute $F(x_1, \ldots, x_)$ for some public Boolean function F known to all parties. To that purpose, the parties can interact together by means of private channels, and they can do local computation. In the end of the protocol each party should terminate with the correct result of the function evaluation.

We say that a SFE protocol for n parties realizes a function F on a common input x if at the end of the execution of the protocol each party $i \in [n]$, who starts the protocol with a private input x_i, gets the output of $F(x, x_1, \ldots, x_n)$.

We say that an adversary controls t parties if it can intercept the messages directed to the private channels of a subset of cardinality t of the n parties. Intuitively, a SFE protocol realizes a function F with (t, n)-privacy (or security) if it realizes F with n parties and any adversary *passively* controlling at most t parties cannot leak any information about the secret inputs of the parties not under its control. Here, by passive control we mean that the adversary can only intercept but not modify the messages sent over the channels.

We formalize the previous concepts with the following definitions.

First, we formalize what a SFE protocol is. Intuitively, a protocol for n parties is computer program, called the *next function*, that is used in each round of the protocol by a specific party $P_i, i \in [n]$ in the following way: given the current local state of the party (consisting of the common input, the secret input of the party, the randomness of the party, and the messages sent and received so far in the previous rounds), the next function outputs the $n-1$ messages directed to every other party in the next round. Precisely, we have the following definition.

Definition 54 (SFE protocol) *A protocol Π for n parties is specified in term of the next message function. That is, given $(i, x, x_i, r_i, (m_1, \ldots, m_j))$, where $i \in [n]$ is the index of party P_i, r_i denote the random coins used by party P_i during the execution of the protocol, $m_1, \ldots, m_j$ denote the messages that the party P_i received in the first j rounds, the next message function outputs the set of $n-1$-messages that the party P_i has to send to every other party in round $j+1$.*

Now, we define the view of a party in the execution of a protocol.

Definition 55 *We define the view, in symbols View_i, of a party $P_i, i \in [n]$ in a protocol Π for n parties to constitute her private input x_i, randomness r_i and the set of messages received and sent by the party during the execution of the protocol.*

For a set $T \subseteq [n]$ of parties in a protocol for n parties, we define View_T be the union of all views of all parties in the set T.

Finally, we define what means for a protocol to realize a function and what means to realize it privately.

Definition 56 *We say that a protocol* Π *for* n *parties realizes a function* F *if for all common inputs* x *and private inputs* $x1, \ldots, x_n$ *the output of each party* $P_i, i \in [n]$ *is equal to* $F(x, x_1, \ldots, x_n)$ *with probability* 1 *where the probability is over the random inputs of each party.*

Definition 57 *We say that a protocol* Π *for* n *parties realizes a function* F *with* (t, n)*-privacy (or security) if* Π *realizes* F *and there exists a simulator* Sim *such that for all* $x, x_1, \ldots, x_n, T \subseteq [n]$ *such that* $|T| \leq t$, $\mathsf{View}_T(x, x_1, \ldots, x_n)$ *is distributed identically to* $\mathsf{Sim}(T, x, \{x_i\}_{i \in T}, F(x, x_1, \ldots, x_n))$.

We show how to implement a (t, n)-private SFE protocol realizing any efficiently computable function $\mathrm{n}F$ using SS and assuming that $t < 2n$. We use Shamir's (t, n)-SS instantiated over the field $F \stackrel{\triangle}{=} \mathbb{Z}_p^\star$.

We assume that the function F is implemented as an arithmetic circuit composed of addition and multiplication gates over F. The protocol will be executed by evaluating the circuit gate by gate as in a non-secure evaluation. We assume that each wire is numbered and carries a value, where this value is the value that is assigned by this wire when evaluating the circuit (in a non-secure way).

At each wires of the circuit (including the input wires) we want to keep the following invariant.

- Input wires.

  For an input wire a with value x, we want to keep the invariant that each party P_i holds a share $p(i)$ of a degree t polynomial such that $p(0) = x$.

- Intermediate wires that are output of an addition gate. If c is the output of an addition gate, let a, b be the wires in input to the gate and let be resp x, y the values assigned to these wires. We want that each party P_i holds a share $p(i)$ of a degree t polynomial such that $p(0) = x + y$. If c is the output of a multiplication gate, let a, b be the wires in input to the gate and let be resp x, y the values assigned to these wires. We want that each party P_i holds a share $p(i)$ of a degree t polynomial such that $p(0) = x \cdot y$.

- Intermediate wires that are output of a multiplication gate. Let us consider a multiplication gate with input wires number a and b and output wire number c. By the invariant, we know that there are two degree t polynomials h_a and h_b such that (1) $h_a(0) = x$ and (2) $h_b(0) = y$, for some values x, y in the field.

Notice that if the invariant holds for each wire, at the last wire of the circuit, the parties can use the reconstruction procedure of Shamir's SS to reconstruct the output of the wire that, by induction, will correspond the correct output of the circuit.

Let us show how to guarantee the invariant.

- Guaranteeing the invariant for input wires. In this case, the value x corresponding to the wire is held by some party P_j who can act as a dealer in Shamir's SS to send the shares $p(i)$ to each party P_i for a degree t polynomial p such that $p(0) = x$. So the invariant is guaranteed.

- Guaranteeing the invariant for addition gates. If c is the output of an addition gate, let a, b be the wires in input to the gate and let be resp x, y the values assigned to these wires. Recall that we want that each party P_i holds a share $p(i)$ of a degree t polynomial such that $p(0) = x + y$.

  By the invariant there exist degree t polynomials g_a, g_b such that $g_a(0) = x, g_b(0) = y$ and each party P_i holds shares $g_a(i), g_a(i)$. We hold the invariant as follows. Each party P_i locally outputs the values $h(i) \stackrel{\triangle}{=} g_a(i) + g_b(i)$. We claim that the $h(i)$'s are the shares of a degree t polynomial h_c such that $h_c(0) = x + y$. Let $h_c(x) \stackrel{\triangle}{=} g_a(x) \cdot g_b(x)$. Observe that h_c has degree t, and moreover $h_c(0) = g_a(0) + g_b(0) = x + y$ and that $h_c(i) = h(i)$, as we had to show.

- Guaranteeing the invariant for multiplication gates.

  Let us consider the following polynomial (3) $h(x) \stackrel{\triangle}{=} g_a(x) \cdot g_b(x)$.

  By (1),(2) and (3) We know that (4) $h(0) = g_a(0) \cdot g_b(0) = x \cdot y$.

  Moreover, $h(x)$ has degree $2t$. Since $2t < n$, by Lagrange's interpolation, (5) there exist values $\lambda_1, \ldots, \lambda_n \in F$ such that $h(0) = \sum_{i \in [n]} \lambda_i \cdot h(i) =$ (by (4)) $= x \cdot y$.

  For any $i \in [n]$, the party P_i computes as follows, P_i generates a random poly H_i of degree t such that (6) $H_i(0) = h(i)$. Furthermore, for any $j \in [n]$, P_i sends the value $H_i(j)$ to party j.

  Observe that for any $i \in [n]$ party P_i receives values $H_1(i), \ldots, H_n(i)$. Finally, each party P_i locally outputs for the wire number c the value $h_{c,i} \stackrel{\triangle}{=} \sum_{j \in [n]} \lambda_j \cdot H_j(i)$, where the values λ_j's are from (5).
  We claim the following: the values $h_{k,1}, \ldots, h_{k,n}$ are shares of a degree t poly $h_c(x)$ such that (7) $h_c(i) = h_{k,i}$ and (8) $h_c(0) = x \cdot y$.

  We prove (7) and (8).

  Let $h_c(x) \stackrel{\triangle}{=} \sum_{j \in [n]} \lambda_j \cdot H_j(x)$.

  Notice that $h_c(i) \stackrel{\triangle}{=} \sum_{j \in [n]} \lambda_j \cdot H_j(i)$ and these values $= h_{c,i}$, so (7) is proven.

  Finally,

$$h_c(0) \stackrel{\triangle}{=} \sum_{j \in [n]} \lambda_j \cdot H_j(0) =$$

  (by (6))

$$\sum_{j \in [n]} \lambda_j \cdot h(j) =$$

(by (5)) $= x \cdot y$. So (8) is proven.

Semi-honest vs Fully Malicious security. The above protocol is secure only against passive adversaries. This level of security is called *semi-honest* security as opposed to *fully malicious* security. For this reason it cannot be effectively used as building block for our e-voting tasks. Moverover, being the protocol interactive it does not achieve universally verifiable, that is people who did not participate in the protocol would not be convinced that the output of the protocol is the correct one.

5 Proofs of knowledge and Digital Signatures

Consider the following problem. Let $\mathbb{G}$ be a group of prime order q and let g, h be two elements of $\mathbb{G}$. Then, there exists $w \in \mathbb{Z}_q$ such that $h = g^w$. However, having g, h does not imply *knowledge* of w.

Someone could have generated g, h without knowledge of w or could have received g, h from another party. On the other hand, if Alice generates $w \leftarrow \mathbb{Z}_q$ and sets $h \stackrel{\triangle}{=} g^w$, it is clear that Alice knows w.

In this case, Alice could be interested in proving to Bob that she knows w. The trivial way to accomplish this is to reveal w. Doing that in ZK instead makes the problem harder and interesting.

Observe that this cannot be expressed as a decision problem. The NP language associated to polynomial-time relation that consists of pairs of the form (g, h) consists of all group elements h such that there exists w such that $h = ^w$, but since the group has prime order, the language consists of all elements in $\mathbb{G}$ and so it is easily decidable.

So, in this case verifier is not interested in discovering whether the instance belongs to the language (since the witness always exists for all group elements) but that the prover actually has knowledge of the witness. This can be formalized as follows.

Definition 58 (Proof of knowledge) *A pair $(\mathcal{P}, \mathcal{V})$ of PPT interactive machines is called a* proof of knowledge *(PoK) with* knowledge error $k(\cdot)$ *for polynomial-time relation $\mathcal{R}$ if completeness and the following property (that is a strengthening of soundness) hold.*

- Knowledge Soundness: *There exists a probabilistic oracle machine* $\mathsf{Extract}$*, called the* knowledge extractor*, such that for every interactive machine $\mathcal{P}^\star$ and for every input x accepted by $\mathcal{V}$ when interacting with $\mathcal{P}^\star$ with probability $\epsilon(|x|) > k(|x|)$, $\mathsf{Extract}^{\mathcal{P}^\star(x)}$ outputs a witness w for x. Moreover, the expected number of steps performed by* $\mathsf{Extract}$ *is bounded by $p(|x|)/(\epsilon(|x|) - k(|x|))^d$, for some polynomial $p(\cdot)$ and constant d.*

The knowledge error $k(\cdot)$ can be thought as the probability that one can convince the verifier without knowing a witness.

It turns out that a Σ-protocol with challenges of length $t(\cdot)$ is a PoK with knowledge error $2^{-t(\cdot)}$.

Theorem 15 *Let Σ be a Σ-protocol for polynomial-time relation $\mathcal{R}$ with challenge length $t(\cdot)$. Then Σ is a PoK with knowledge error $2^{-t(\cdot)}$.*

The idea of the proof is that an extractor can rewind the prover on the same first message but different challenges so, by the simulation soundness, is able to extract the witness.

By rewinding we mean that the extractor will first run the prover with same random tape and will get from it a first message a, then will simulate to it the role of the verifier by sending e_1 and waiting for the final reply z_1. Then, the extractor will run again the prover on the same random tape, so the extractor will get again the same first message a but this time will send the prover a different challenge e_2 and will wait for the final reply z_2. If the two transcripts (a, e_1, z_1) and (a, e_2, z_2) are accepting for the verifier, the extractor will be able to extract the witness (using the other extractor guaranteed by the simulation soundness).

The entire proof of the theorem is more complicated since one has to take in account that the prover convince the verifier with some probability $\epsilon(\cdot)$. We defer the full proof to `https://cs.au.dk/~ivan/Sigma.pdf`.

We now show a protocol for proving knowledge of discrete logs, that is for proving knowledge of the fact that $h = g^w$ for some w. This protocol will be used as a basis for adding several features and security properties to our toy e-voting scheme.

Henceforth, we assume $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda>0}$ to be a family of groups of prime order q_λ that are in the range of some generator Gen of families of groups.

Definition 59 ($\mathcal{R}_{\mathsf{DLOG}}$) *The relation of discrete log pairs (for the family of groups $\mathcal{G}$) is defined as follows: $\mathcal{R}_{\mathsf{DDH}}((\mathbb{G}, q, g, h), w) = 1$ iff $\mathbb{G} \in \mathcal{G}$, $g, h \in \mathbb{G}$ and $h = g^w$ for some non-negative integer $w < \mathbb{Z}_q$. It is easy to see that this is a polynomial-time relation.*

We now show a Σ-protocol for $\mathcal{R}_{\mathsf{DLOG}}$.

Definition 60 (Σ_{DLOG}) *Σ_{DLOG} consists of the following pair of interactive machine $(\mathcal{P}, \mathcal{V})$. Let L be the language associated with $\mathcal{R}_{\mathsf{DLOG}}$.*

- *Inputs. $\mathcal{P}$ and $\mathcal{V}$ start with the common input $x \stackrel{\triangle}{=} (\mathbb{G}, q, g, h) \in L$ and $\mathcal{P}$ additionally takes as input the witness $w \in \mathbb{Z}_q$.*
- *First message. $\mathcal{P}$ chooses a random $r \in \mathbb{Z}_q$ and sends $a \stackrel{\triangle}{=} g^r$ as first message.*
- *Challenge. $\mathcal{V}$ sends a random element $c \leftarrow \mathbb{Z}_q$ as its challenge.*
- *Response. $\mathcal{P}$ responds to the challenge sending $z \stackrel{\triangle}{=} r + c \cdot w \mod q$.*

- *Verifier's decision.* $\mathcal{V}$ *on input* x*, the first message* a*, the challenge* c *and the response* z *outputs* 1 *iff*

$$g^z = a \cdot h^c.$$

Observe that the protocol can be seen as a simplification of the Σ-protocol for $\mathcal{R}_{\mathsf{DDH}}$ and as such we omit the proof of the following theorem.

Theorem 16 Σ_{DLOG} *is a* Σ*-protocol for* $\mathcal{R}_{\mathsf{DLOG}}$*.*

5.1 Digital Signatures

Digital signatures are the digital version of handwritten signatures. A valid digital signature gives a recipient guarantee that the message was created by a known sender, and that the message was not altered in transit. The sender is identifed by a verification key (also called public key) and the sender is able to sign a message using his own secret key. We have the following formalization.

Definition 61 (Digital Signature scheme) *A digital signature (DS) scheme* $\mathsf{DS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verfy})$ *is a tuple of* 3 *PPT algorithms.*

- $\mathsf{Setup}(1^\lambda)$*: on input the security parameter* λ *output* verification key VK *(also called the public key) and* secret key SK*.*
- $\mathsf{Sign}(\mathsf{SK}, M)$*: on input secret key* PK *and a message* $M \in \{0,1\}^\star$*, outputs* signature σ*.*
- $\mathsf{Verfy}(\mathsf{VK}, \sigma, M)$*: on input the verification key* VK *for security parameter* λ*, a signature* σ*, and a message* $M \in \{0,1\}^\star$*, output* 0 *(reject) or* 1 *(accept).*

A DS scheme has to satisfy the following properties.

- *(perfect) correctness*, that is that for all $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $M \in \{0,1\}^\star$, for all $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, M)$, $\mathsf{Verfy}(\mathsf{VK}, \sigma, M) = 1$.
- *Unforgeability*: We denote by $\mathsf{A}^{\mathsf{Sign}(\mathsf{SK},\cdot)}$ an adversary that has oracle access to the signing algorithm $\mathsf{Sign}(\mathsf{SK}, \cdot)$, This means that the adversary can query the oracle on any message M and get as answer $\mathsf{Sign}(\mathsf{SK}, M)$ and the oracle query will count as one step. Notice that the adversary does not see the secret key. In the following, given an adversary $\mathsf{A}^{\mathsf{Sign}(\mathsf{SK},\cdot)}$ we denote by O^{A} the set of all oracle queries that $\mathsf{A}^{\mathsf{Sign}(\mathsf{SK},\cdot)}$ made to its oracle.

  For every nuPPT adversary $\mathsf{A}^{\mathsf{Sign}(\mathsf{SK},\cdot)} \stackrel{\triangle}{=} \{\mathsf{A}_\lambda^{\mathsf{Sign}(\mathsf{SK},\cdot)}\}_{\lambda>0}$ the following function is negligible in λ:

  $$\mathrm{Prob}[\mathsf{A}_\lambda^{\mathsf{Sign}(\mathsf{SK},M)}(\mathsf{VK}) = (\sigma^\star, M^\star) \wedge M^\star \notin O^{\mathsf{A}_\lambda} \wedge \mathsf{Verfy}(\mathsf{VK}, \sigma^\star, M^\star) \mid (\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda)].$$

5.1.1 From proof of knowledge to digital signature

The FS transform that we saw before to convert interactive ZK proofs into NIZK can be also applied to convert a PoK into a DS scheme.

First, we need to introduce the concept of a *hard relation.*

Definition 62 (Hard relation) *A polynomial-time relation $\mathcal{R}$ associated with a language L is said to be hard with respect to an algorithm* Gen *(called the generator) if:*

- Gen*, on input* 1^λ*, outputs a pair* $(x, w) \in \mathcal{R}$ *where* $|x| = \lambda$*.*
- *For all nuPPT algorithms* $\mathsf{A} \stackrel{\triangle}{=} \{A_\lambda\}_{\lambda>0}$*, the quantity* $\epsilon(\lambda) \stackrel{\triangle}{=} \text{Prob}[(x, w) \in \mathcal{R} |\ x \leftarrow \mathsf{Gen}(1^\lambda);\ w \leftarrow \mathsf{A}_{|x|}(x)]$ *is a negligible function in* λ*.*

A polynomial-time relation $\mathcal{R}$ associated with a language L is said to be hard if it is hard with respect to some PPT algorithm Gen*.*

A hard relation is essentiall one in which given an instance x it is difficult to find a witness for x. It can be seen that the relation $\mathcal{R}_{\mathsf{DLOG}}$ we saw before is hard under the discrete log assumption.

Let $\Sigma \stackrel{\triangle}{=} (\mathcal{P}, \mathcal{V})$ be a PoK for a hard polynomial-time relation $\mathcal{R}$. Recall that a transcript of Σ has the form (a, c, z). The DS scheme $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verfy})$ built from Σ would work as follows.

The setup algorithm Setup will run the generator of the hard relation to compute a pair $(\mathsf{VK}, \mathsf{SK})$ where VK is the instance of the relation and SK is the witness to it.

The signing algorithm Sign on input SK and $M \in \{0,1\}^\star$ runs $\mathcal{P}$ to get the first message a, then sets $c = H(\mathsf{VK}, a, M)$, where H is an hash function modelled as a RO, and sends c to $\mathcal{P}$ and gets z as answer. Finally, the signing algorithm outputs $\sigma \stackrel{\triangle}{=} (a, c, z)$ as signature of M.

The verification algorithm Verfy on input verification key VK, signature $\sigma \stackrel{\triangle}{=} (a, c, z)$ and message $M \in \{0,1\}^\star$, it verifies that (1) $c = H(\mathsf{VK}, a, M)$ and (2) (a, c, z) is an accepting transcript for instance VK.

We defer the details of the analysis of the above DS. The intuition is that, in order to forge signatures, the attacker should break the PoK property. The above scheme can be instantiate for instance with the Σ-protocol Σ_{DLOG} for $\mathcal{R}_{\mathsf{DLOG}}$.

Fourth iteration of our toy e-voting scheme. To solve the issue of eligibility, we assume that each voter V is registered in the system, that is the verification key VK_V of V is made public and is known by anyone including the authorities and V retains his secret key SK_V. When V submits his ciphertext CT to the PBB, V adds VK_V and a signature σ of B under VK_B.

When tallying the authorities discard each ciphertext CT for which the corresponding signature σ is not valid with respect to the corresponding verification

key VK attached to the ciphertext. Moreover, the authorities discard all ciphertexts corresponding to the same verification key in order to identify attempts of *double voting*, and discard all equal ciphertexts to identy attempts to copy and paste attacks.

To withstand re-randomization attacks, we require a voter to add a proof of knowledge of the element $\mathsf{CT}_1 = g^r$ of his ciphertext $\mathsf{CT} \triangleq (\mathsf{CT}_1, \mathsf{CT}_2)$. In this way, if an adversary attempts to produce a re-randomization $\mathsf{CT}' = (\mathsf{CT}'_1, \mathsf{CT}'_2)$ of a ciphertext CT, it will not be able to generate a valid PoK of CT'_1. The authorities will then discard each ciphertext with invalid PoK.

The rest of the scheme is unchanged. In this way, only eligible voters can vote and they can only vote once, and copies of a voter's vote are not allowed.

The last iteration of our toy e-voting scheme still suffers from the following deficiency. Suppose the first $m-1$ authorities setup their PKs $\mathsf{PK}_1, \ldots, \mathsf{PK}_{m-1}$ honestly but the m-th authority sets $\mathsf{PK}_m = \mathsf{PK}' \cdot \prod_{i=1}^{m-1} \mathsf{PK}_i^{-1}$, where PK' is some PK for which the m-th authorithy knows the corresponding SK SK. In this way, the general PK can be seen to be $\mathsf{PK} \triangleq \mathsf{PK}'$ and the m-th authority has complete control over the privacy of the individual votes. To address this issue, in Section 6.4 we will introduce the notion of commitment scheme.

6 Commitments

A commitment scheme is a protocol between a Sender and a Receiver. The Sender holds a message m and, in the first phase, it picks a randomness r and then "encodes" the message using r and sends the encoding (a commitment to m) to the Receiver. In the second phase, the Sender opens the commitment by sending to the Receiver m and the randomness r so that the Receiver can verify that the content of the commitment was the message m. A commitment scheme should satisfy two security properties:

- Hiding. Receiving a commitment to a message m should give no information to the Receiver about m.
- Binding. The Sender cannot "cheat" in the second phase and convince the Receiver that the content of the commitment was a message m' different from m.

It is impossible to satisfy both properties against computationally unbounded adversaries. It is possible, however, to have schemes in which the Hiding property holds against computationally unbounded Receivers and the Binding property holds (under appropriate assumptions on the primitive used in the construction) against nuPPT Senders; and it is possible to have schemes in which the Hiding property holds against nuPPT Receivers and the Binding property holds against unbounded Senders.

6.1 Commitments from any PKE

Any PKE $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with perfect correctness is a computationally hiding and perfectly binding commitment scheme. To commit to a message m, the Sender computes $(\mathsf{PK}, \mathsf{SK}) = \mathsf{Setup}(1^\lambda; r_s)$, choose randomness r_e, and compute $\mathsf{CT} = \mathsf{Enc}(\mathsf{PK}, m; r_e)$ and outputs $(\mathsf{PK}, \mathsf{CT}, r_s, r_e)$ as commitment of m. To open the commitment, the Sender reveals to the Receiver the randomness r_s, r_e and m and the Receiver can run the setup and encryption algorithms on resp. r_s, r_e to verify that CT is encryption of m.

The perfect bindingness property follows from the perfect correctness and the computational hiding property follows from the $\mathsf{IND\text{-}CPA}$ security of PKE.

6.2 Commitments from hash functions

Suppose H is an hash function modelled as a RO. To commit to a message m, the Sender chooses random string r of a certain length n (the length n affects the security of the system) and sends the commitment $c = H(m, r)$ to the Receiver. To open the commitment, the Sender reveals to the Receiver the randomness r and the Receiver can compute $c' = H(m, r)$ and check that $c' = c$.

The computational bindingness property holds in the RO model noticing that a PPT adversary can make at most a polynomial number of queries $p(n)$ to RO. So it can be seen that the probability of finding two valid openings is negligible in n. The computational hiding property follows similarly in the RO model.

6.3 Homomorphic commitments

Pedersen commitments are built from a secure group $\mathbb{G}$ of prime order q. Suppose there are public parameters $g, h \in \mathbb{G}$.

The Sender commits to message m by choosing randomness r and sending $c = g^m \cdot h^r$ to the Receiver. The values (g, h) can be also seen as public generators that are known by all participants or can be setup by the Receiver in an initial phase. Moreover, they can be used for many commitments (not just for the commitment to a single message). The opening of c consists of m and r by means of which the Receiver can compute by himself $c' = g^m \cdot h^r$ and verify that $c' = c$.

Pedersen's commitments are perfectly hiding: given a commitment c and two pairs of messages m, m' there exist two random values r, r' such that $c = g^m \cdot h^r = g^{m'} \cdot h^{r'}$. In fact, let $h = g^z$, for some $z \in \mathbb{Z}_q$ and let $m' = m + d \mod q$ for some $d \in \mathbb{Z}_q$. Then $c = g^m \cdot g^{zr} = g^{m'+d+z \cdot r} = g^{m'} \cdot h^{r+d/z}$, so $r' = r + d/z \mod q$.

Pedersen's commitments are computationally binding assuming the discrete log assumption: suppose that there exists a PPT adversary A that, given a commitment c is able to find two pairs (m, r) and (m', r') such that $c = g^m \cdot h^r = g^{m'} \cdot h^{r'}$, and $m \neq m'$. Let $h = g^z$, for some $z \in \mathbb{Z}_q$ and let $m' = m + d \mod q$ for some $d \in \mathbb{Z}_q$. Then, it follows that $g^{d/(r-r')} = g^z$, so $z = d/(r - r') \mod q$

is a solution to the equation $h = g^z$. Therefore, such adversary A can be used to build an adversary B that breaks the discrete log assumption.

Perdersen's commitments are homomorphic: if c_1, c_2 are resp. commitments to messages m_1, m_2 then $c_1 = g^{m_1} \cdot h^{r_1}, c_2 = g^{m_2} \cdot h^{r_2}$, and so $c_1 \cdot c_2 = c_1 = g^{m_1+m_2} \cdot h^{r_1+r_2}$ is a valid commitment to message $m_1 + m_2 \mod q$ with randomness $r_1 + r_2$.

Pedersen's commitments are *trapdoor* commitments. This means that if the Sender knows the value z such that $h = g^z$, the Sender can then break the binding property at his like. In fact, from the analysis of the binding property it can be seen that if z is known then r' can be chosen so as to satisfy the equation $r' = r - d/z$. For this reason, the values g, h must be chosen by the Receiver or be public parameters not under the control of the Sender.

6.4 Commitments from Σ-protocols

Let $\mathcal{R}$ be a hard polynomial-time relation. We now show how to build a commitment scheme from a Σ-protocol $\Sigma = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$. Let Sim be the simulator guaranteed by the HVZK property for Σ and let t be the challenge length of Σ.

To commit to a string $e \in \{0,1\}^t$, Receiver samples a hard pair (x, w) from $\mathcal{R}$, and sends x to the Sender. Sender runs Sim on (x, e) to get (a, e, z) and sends a to the Receiver.

To open a commitment a, the Sender sends (a, e, z) to the Receiver who verifies that (a, e, z) is an accepting transcript for x.

By the HVZK property, a is independent of e. In fact, simulated proofs are distributed identically to honestly generated proofs in which, in turn, the first messages are independent from the challenges. Thus, the hiding property follows.

The binding property follows from the fact that opening to two e, e' for the same a implies, by special soundness, finding a witness w to x.

Last iteration of our toy e-voting scheme. To solve the issue present in the previous iteraion of our toy e-voting scheme, namely the fact that one authority can create its own PK so as to know the SK corresponding to the general PK, we employ commitments in the following way. Each authorithy $A_i, i \in [m]$ commits to its PK PK_i sending the corresponding commitment c_i to the PBB.

Only when all m commitments are published on the PBB, the m authorities open the commitments, and everyone can see the m PKs $\mathsf{PK}_1, \ldots, \mathsf{PK}_m$ and then the protocol proceeds as before. Notice that in this way the attack is prevented because none of the authorities can base its PK on the others.

6.5 ZK from SFE and commitments

We now show a construction of a ZK IPS for any polynomial-time relation $\mathcal{R}$ from a (semi-honest) SFE protocol Π for a related function and a statistically

binding commitment scheme. The parameters needed for Π will be specified next. The resulting ZK IPS can be turned into a NIZK in the ROM using the FS transform.

We first need the following definitions.

Definition 63 *We say that two views* View_i *and* View_j *of parties* P_i *and* P_j, $i, j \in [n], i \neq j$ *are consistent with respect to a SFE protocol* Π *for* n *parties and common input* x *if the set of messages sent from* P_i *to* P_j *is exactly equal to the set of messages received by party* P_j *from* P_i *and vice versa.*

Theorem 17 *Let* $\mathsf{View}_1, \ldots, \mathsf{View}_n$ *be a set of (possibly incorrect)*[11] *views of* n *parties. Let* Π *be a SFE protocol for* n *parties and* x *be a common input. Then all pairs of views* $\mathsf{View}_i, \mathsf{View}_j$ *are consistent with respect to* Π *and* x *if and only if there exists an honest execution of* Π *with common input* x *and some choice of private inputs* $x_1, \ldots, x_n$ *and random inputs* $r_1, \ldots, r_n$ *in which* View_i *is the view of party* P_i *for every* $i \in [n]$.

Proof 14 *The "if" direction follows directly from the definitions. For the "only if" direction, let* $\mathsf{View}_1, \ldots, \mathsf{View}_n$ *be pairwise consistent views and for any* $i \in [n]$ *let* $x_i r_i$ *be the secret input and random input reported in view* View_i. *The pairwise consistency of the views implies, by induction on the number of rounds* d, *that the actual view of every player* P_i *after* d *rounds when* Π *is invoked on* $(x, (x_1, r_1), \ldots, (x_n, r_n))$ *is the same as the view of* P_i *in the first* d *rounds reported in* View_i. *It follows that the* n *views* $\mathsf{View}_1, \ldots, \mathsf{View}_n$ *are consistent with the full execution of* Π, *as required.*

Let $\mathcal{R}$ be a polynomial-time relation. Let Π be a SFE protocol for n parties that realizes the function $F(x, w_1, \ldots, w_n) \stackrel{\triangle}{=} \mathcal{R}(x, \oplus_{i=1}^n w_i)$. We will also use a statistically binding commitment scheme.

We now show a ZK IPS protocol $(\mathcal{P}, \mathcal{V})$ for a $\mathcal{R}$.

Definition 64 (ZK protocol from SFE) *The ZK IPS protocol consists of the following pair of interactive machine* $(\mathcal{P}, \mathcal{V})$. *Let* L *be the language associated with* $\mathcal{R}$.

- *Inputs.* $\mathcal{P}$ *and* $\mathcal{V}$ *start with the common input* $x \in L$ *and* $\mathcal{P}$ *additionally takes as input the witness* w *such that* $\mathcal{R}(x, w) = 1$.
- *First message.* $\mathcal{P}$ *chooses random* $w_1, \ldots, w_n$ *subject to the condition that* $\oplus_{i=1}^n w_i = w$.

  $\mathcal{P}$ *runs the protocol* Π *"in his head" (by choosing uniform random coins for each party) to generate the views of each party* P_i.

  Let View_i *denote the view of party* P_i *in the execution of* Π.

  $\mathcal{P}$ *generates commitment to each* View_i *separately using a statistically binding commitment scheme and sends the commitment to* $\mathcal{V}$.

[11] That is, such views do not necessarily come from an honest execution of Π on some set of inputs.

- *Challenge.* $\mathcal{V}$ *chooses random* $i, j \in [n]$ *and sends it to* $\mathcal{P}$.
- *Response.* $\mathcal{P}$ *sends to* $\mathcal{V}$ *the openings of the commitments to the views* View_i *and* View_j.
- *Verifier's decision.* $\mathcal{V}$ *accepts iff the openings are correct, the views* View_i *and* View_j *are consistent and the output of parties* P_i, P_j *in such views is both* 1.

For simplicity, we just prove that the previous protocol is HVZK; the proof for (full) ZK is more intricated and can be found at [`https://web.cs.ucla.edu/~rafail/PUBLIC/77.pdf`]

Theorem 18 *If the protocol* Π *is a SFE protocol thar realizes* F *with* $(2, n)$*-privacy then the protocol* $(\mathcal{P}, \mathcal{V})$ *satisfies HVZK and* $\left(1 - \frac{1}{\binom{n}{2}}\right)$*-soundness.*

Proof 15 *The correctness guarantee is easy to verify. We now argue soundness and ZK. Suppose* x *is not in the language* L*. Then, for every* $w_1, \ldots, w_n$ *it holds that* $F(x, \oplus_{i=1}^n w_i) = 0$*. If the prover executes the protocol* Π *honestly then by the fact that* Π *realizes* F*, it follows that for all choices of random coins generated by the prover, the protocol* Π *outputs* 0 *(each party in the protocol will output* 0*). If the prover tries to cheat then it means that the prover does not execute the protocol honestly. By Thm 17, it follows that there exist* $i, j \in [n], i \neq j$ *such that the views* View_i *and* View_j *are inconsistent.*

Therefore, the verifier $\mathcal{V}$ *can catch a cheating prover with probability* $\frac{1}{\binom{n}{2}}$.

We now show HVZK. Consider the following simulator Sim*. The simulator chooses* i, j *uniformly at random under the condition that* $i \neq j$*.* Sim *then chooses random* w_i *and* w_j *and runs the simulator guaranteed by the* $(2, n)$*-privacy of* Π *to obtain views* View_i *and* $View_j$*. It then commits to* $View_k$ *equal to the strings of all 0s, for all* $k \in [n].k \neq \{i, j\}$ *and commits to* View_i *and* View_j.

The view of Sim *is identical to the view of* $\mathcal{V}$ *from the* $(2, n)$*-privacy condition of* Π.

Observe that the previous protocol is Σ-protocol and as such can be turned into a NIZK using the FS transform.

7 Verifiable Shuffles

We observe that our toy e-voting scheme based on the homomorphic properties of ElGamal is well suited for referendums, that is elections in which each voters can cast either of two options, but it is not straightforward how to extend the construction to support general types of elections in which the voters can cast more complex preferences and the tally is a complex function of the individual preferences.

In this Section we propose an alternative paradigm to construct e-voting schemes to address these issues.

The basic idea is that the voters still submits their ballots as encryption of their preference using a threshold PKE scheme. Then, each ciphertext is decrypted by the authorities to get each individual preference in the clear and from all the individual preferences the tally can be computed. Contrast this with our e-toy voting scheme of Section 6.4 in which, using the homomorphic properties of ElGamal, the ballots of the voters are used to compute a single ciphertext encrypting the tally of the election and then only such ciphertext is decrypted by the authorities. Observe that in this way we do not need to assume and use any homomorphic property from the encryption scheme.

The problem is that each ciphertext needs to be sent along with a digital signature that identifies a voter. While, the public-key may be a pseudonym in the sense that it does not directly contain information about the voter, this is in practice a very weak anonymization guarantee. For instance, the public-key needs to be sent from some authority to the voter either in online or in physical form. So there exists some authority who knows the association between each public-key and the voter.

Yet, a voter can use the secret-key corresponding to the same public-key to sign other online documents. Observe also that a public-key could be used in different elections so it is possible to identify which voters changed preferences along the time.

Furthermore, there is another issue that would allow the adversaries to break the privacy of the voters. A voter needs to send her ciphertext to the PBB and this is done using the TCP/IP protocol on which the current Internet is based. In any manner the PBB is implemented, there needs to be some computer over the Internet who will receive the ballot (i.e., the ciphertext) of the voter. Such computer will know the IP address of the sender of the ballot. An IP address is another kind of pseudonym but it is more prone to de-anonymization attacks. Indeed, notice that people may use the same IP address to connect to different computers and so leave traces of their identities over the Internet. For instance, a voter can submit her ballot to the PBB using a certain IP address and with the same IP address can later write on a public forum using her real identity. If an attacker, who can see the IP address of such voter, is also able to find occurrences of her IP address on the forum or elsewhere, the attacker will be able to de-anonymize the voter.

Also, if an attacker, who can see the IP addresses of a voter V who submit her ballots to the PBB, colludes with the Internet service provider of V, then the attacker will be able to de-anonymize V.

For these reasons, pseudonymity is not enough and we need a way to break the link between the ballots and the public-keys and the IP addresses of the voters.

This can be done using one of the two following techniques. We only sketch the first one since it will be the second approach on which we will focus.

- Technique I: Shuffle from re-randomizable encryption. Let us assume that

the ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ resp. encrypting the preferences $v_1, \ldots, v_n$ of the voters $V_1, \ldots, V_N$ are published on the PBB along with the pairs public-key and signature $(\mathsf{PK}_1, \sigma_1), \ldots, (\mathsf{PK}_n, \sigma_n)$). We also assume that the ciphertexts are generated using a threshold PKE scheme that is also *re-randomizable* as ElGamal (any other threshold PKE scheme that is re-randomizable suffices). The public-keys and the signatures can be used to verify that such ballots come from eligible voters. Then, the following protocol can be applied.

We assume that there are d *mixers* $M_1, \ldots . M_d$. The mixers may also be the same authorities that perform threshold decryption. At the beginning the list L_1 is filled with the ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ of the n voters encrypting resp. votes $v_1, \ldots, v_n$. For $i = 1, \ldots, d$, the mixer M_i (that is also the d-th authority) performs the following operations:

- The mixer M_i re-randomizes each ciphertext $\mathsf{CT}_j, j \in [n]$ using a new random value $r_{i,j}$ to compute a new ciphertext CT'_j encrypting the same plaintext v_j and sets the list $L'_i = (\mathsf{CT}'_1, \ldots, \mathsf{CT}'_n)$.
- Choose a random permutation γ_i from $[n]$ to $[n]$ and compute the list $L''_i = (\mathsf{CT}'_{\gamma_1(1)}, \ldots, \mathsf{CT}'_{\gamma_1(n)})$. Publish L''_i on the PBB. Set L_{i+1} to L''_i. The new list L_{i+1} will be the input to the next mixer.

In the end, the authorities can decrypt each ciphertext in the list L_d to get the n individual preferences in a permuted way and from them the tally can be computed.

Observe that even if an attacker knew that $\mathsf{CT}_{1,j}, j \in [n]$ comes from a certain IP address X, the attacker will not know which is the ciphertext in a list $L_i, i \in [n]$ that encrypts v_j, unless the attacker gets from the mixers $M_1, \ldots, M_i$ the randomness vectors $(r_{k,j}, \ldots, r_{k,n}), k = 1, \ldots, i$ used to re-randomize the original ciphertexts and the permutations $\gamma_1, \ldots, \gamma_i$. As byproduct, if the attacker is not able to collude all the mixers, the attacker will not be able to de-anonymize the voters.

- Technique II: Shuffle from malleable encryption.

  Here, the mixers are the same authorities that perform the partial decryption and we assume that the threshold encryption scheme is malleable. We assume the message space of the encryption scheme to be a group of prime order q generated by a generator g. There is an efficient *exponentiation operation* that, given a ciphertext CT for message m encrypted with the PK PK corresponding to the SK SK and given a value $c \in \mathbb{Z}_q$, can compute a *mauled* ciphertext CT' that decrypts to m^c. Moreover, no nuPPT adversary can distinguish with more than negligible advantage between (g, g^c, CT) and $g, g^c, \mathsf{CT}')$, where CT is the output of the encryption with input PK and m and CT' is computed by first setting CT to be the output of the encryption with input PK and m and then using the exponentiation operation with inputs CT and c.

Notice that ElGamal satisfies this property using the following exponentiation operation: raise each coordinate of an ElGamal ciphertext to the power c. That is, given a ciphertext $(g^r, h^r \cdot m)$[12] for PK h and given a value c the exponentiation operation computes the mauled ciphertext $(g^{rc}, h^{rc} \cdot m^c)$ that decrypts to m^r.

We assume the mixers to be the same authorities that share the secret-keys corresponding to the general public-key of the threshold ElGamal scheme.

At the beginning the list L_1 is filled with the ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ of the n voters. For $i = 1, \ldots, d$, the mixer M_i (that is also the d-th authority) performs the following operations:

- c_i is chosen randomly and independently from $\mathbb{Z}_q$ and commitment $C = g^{c_i}$ is published.
- First, all ciphertexts in the list L_i are exponentiated to the same power c to compute the ciphertexts $\mathsf{CT}'_1, \ldots, \mathsf{CT}'_n$.
- Then, choose a random permutation γ_i from $[n]$ to $[n]$ and computes the list $L'_i = (\mathsf{CT}'_{\gamma_i(1)}, \ldots, \mathsf{CT}'_{\gamma_i(n)})$. The list L'_i is published on the PBB.
- The mixer (one of the authorities) decrypts the ciphertexts in L'_i with its share of the secret-key to output a new list L''_i of partially decrypted ciphertexts that is published on the PBB. L_{i+1} is set to L''_i. The new list L_{i+1} will be the input to the next mixer.

At this point, the messages that are decrypted are the f-th power of the original messages, where $f = \prod_{i \in [d]} c_i$. This is easily fixed by asking each authority in turn to raise the message to the $1/f$-th power.

The only remaining issue is that the mixers could not perform operations correctly. To solve this final problem we need to show how to construct verifiable shuffles, and in particular in the rest of this section we will show a construction for a verifiable shuffle from malleable encryption.

Finally, we will show that the techniques we present can be used to achieve another e-voting application, namely *anonymous e-voting*, see Section **??**.

7.1 The Iterated Logarithm Multiplication Problem

To construct a verifiable shuffle we first build proofs for the iterated logarithm multiplication problem (ILMP). In the ILMP for parameter k, one wants to prove that two tuples $X = (X_1, \ldots, X_k)$ and $Y = (Y_1, \ldots, Y_k)$ are such that $\prod_{i=1}^k \mathbf{dlog}_g X_i = \prod_{i=1}^k \mathbf{dlog}_g Y_i$, where g is a generator of a group of order q. For simplicity, hereafter we consider tuples of 3 elements, that is we consider the ILMP for $k = 3$.

[12] Here, we assume g to generate a group of prime order q.

Observe that ILMP for $k = 2$ corresponds to the problem of proving that a tuple of 4 group elements is DH, so ILMP can be seen as a generalization of the DH problem.

We assume the prover to know the discrete logs in base g of all the elements X_i's and Y_i's.

A Σ-protocol for the ILMP. Consider the following Σ-protocol $\Sigma_{\mathsf{ILMP}} \stackrel{\triangle}{=} (\mathcal{P}, \mathcal{V})$.

Definition 65 (Σ_{ILMP}) *The prover $\mathcal{P}$ chooses random $\Theta_1, \Theta_2 \leftarrow \mathbb{Z}_q$ and send values:*

$$A_1 \stackrel{\triangle}{=} Y_1^{\Theta_1}, A_2 \stackrel{\triangle}{=} X_2^{\Theta_1} \cdot Y_2^{-\Theta_2}, A_3 \stackrel{\triangle}{=} X_3^{\Theta_2}.$$

The verifier $\mathcal{V}$ replies with a random challenge $e \leftarrow \mathbb{Z}_q$.

The prover $\mathcal{P}$ chooses random values $r_1, r_2 \leftarrow \mathbb{Z}_q$ satisfying the following three verification *equations:*

$$Y_1^{r_1} = A_1 \cdot X_1^{-e}, X_2^{r_1} \cdot Y_2^{-r_2} = A_2, X_3^{r_2} = A_3 \cdot Y_3^{-e},$$

and sends r_1, r_2 to $\mathcal{V}$.

Observe that the prover can efficiently do that as it knows the exponents $x_1, x_2, x_3, y_1, y_2, y_3$.

The verifier $\mathcal{V}$ accepts iff all the above three verification equations are satisfied.

We now analyze the above Σ_{ILMP}.

- Completeness. It is easy to see that the completeness holds and the full analysis is left as exercise.
- Soundness. Let us show that the soundness holds.

  Let us set $\bar{r}_i = r_i - \Theta_i, i \in [3]$. Observe that if the values A_1, A_2, A_3 are generated correctly, the first verification equation implies (1) $\bar{r}_1 \cdot y_1 = -e \cdot x_1$, the second verification equation implies (2) $\bar{r}_1 \cdot x_2 - \bar{r}_2 \cdot y_2 = 0$ and the third verification equation implies (3) $\bar{r}_2 \cdot x_3 = -e \cdot y_3$. Replacing (1) and (3) in (2) we have $-e \cdot \frac{x_1 \cdot x_2}{y_1} + e \cdot \frac{y_3 \cdot y_2}{x_3} = 0$ that implies $x_1 x_2 x_3 = y_1 y_2 y_3$ whenever $e \neq 0$. Now suppose that the values A_1, A_2, A_3 are possibly ill-formed. It is easy to see that this means that $A_3 = X_3^{\Theta_2} \cdot g^z$, for some $z \in \mathbb{Z}_q$. Suppose now that $x_1 x_2 x_3 \neq y_1 y_2 y_3$, i.e., the statement is false. Let us set $f \stackrel{\triangle}{=} x_1 x_2 x_3 - y_1 y_2 y_3 \neq 0$.

  The first verification equation implies (1) as before and the second verification equation implies (2) as before but the third verification equation now implies (4) $\bar{r}_2 \cdot x_3 = -e \cdot y_3 + z$. Replacing (1) and (4) in (2) we have $-e \cdot \frac{x_1 \cdot x_2}{y_1} + e \cdot \frac{y_3 \cdot y_2}{x_3} = \frac{z'}{,}$ for some $z' \in \mathbb{Z}_q$ that implies $e \cdot x_1 \cdot x_2 \cdot x_3 = e \cdot y_1 \cdot y_2 \cdot y_3 + z''$, for some z''_q, that is (5) $e \cdot f = z''$. The probability over the choices of e that (5) is verified is $\frac{1}{q}$. Therefore, $\mathcal{V}$ accepts with negligible probability.

- HVZK. Whenever $c, r_1, r_2 \leftarrow \mathbb{Z}_q$, and $A_1 = Y_1^{r_1} \cdot X_1^e, A_2 = X_2^{r_1} \cdot Y_2^{-r_2}, A_3 = X_3^{r_2} \cdot Y_3^e$, the triple $((A_1, A_2, A_3), c, (r_1, r_2))$ is distributed as in a real interaction between $\mathcal{P}$ and $\mathcal{V}$ on the same common input $(X_1, X_2, X_3, Y_1, Y_2, Y_3)$. so, the HVZK property holds.

7.2 A Σ-protocol for proving the correctness of a shuffle

From ILMP we move to the following n-shuffle of group elements problem. Constants $c, d \in \mathbb{Z}_q$ are known to the prover and commitments $C = g^c$ and $D = g^d$ are published. The prover has to convince the verifier that there is some permutation $\gamma : [n] \rightarrow [n]$ such that:

$$Y_i^d = X_{\gamma(i)}^c,$$

for all $i \in [n]$. We assume the prover to know the discrete logs in base g of all the elements X_i's and Y_i's; we will later show how to remove this assumption.

We remark that later on we will only need a version of this protocol in which d can be set to 1 but we present the general version here.

Consider the following interactive protocol $\mathsf{IP}_{\mathsf{Shffl}} \stackrel{\triangle}{=} (\mathcal{P}, \mathcal{V})$ for the n-shuffle of group elements problem.

Definition 66 ($\mathsf{IP}_{\mathsf{Shffl}}$) *The protocol proceeds as follows.*

The verifier sends to the prover $t \leftarrow \mathbb{Z}_q$.

Prover and verifier publicly compute $U \stackrel{\triangle}{=} D^t = g^{dt}$ and $W \stackrel{\triangle}{=} C^t = g^{ct}$ and

$$\bar{X} \stackrel{\triangle}{=} (\bar{X}_1, \ldots, \bar{X}_n) \stackrel{\triangle}{=} (X_1/U, \ldots, X_n/U)$$

and

$$\bar{Y} \stackrel{\triangle}{=} (\bar{Y}_1, \ldots, \bar{Y}_n) \stackrel{\triangle}{=} (Y_1/W, \ldots, Y_n/W).$$

Prover sends a NIZK proof π for the ILMP for the vectors $(\bar{X}, [C]^n)$ and $(\bar{Y}, [D]^n)$, where $[C]^n$ (resp. $[D]^n$) denotes a list containing the value C (resp. D) repeated n times. (Here, we are implicitly assuming to have a NIZK proof for the generalization of the ILMP for parameter $k = 2n$.)

$\mathcal{V}$ accepts iff the proof π is accepted for the above vectors $(\bar{X}, [C]^n)$ and $(\bar{Y}, [D]^n)$.

- Completeness. The analysis of the completeness is straightforward and is left as exercise.

- Soundness. By the soundness of the ILMP proof, if the proof is accepted, except with negligible probability, it holds that:

$$c^n \cdot \prod_{i=1}^{n}(x_i - dt) = d^n \cdot \prod_{i=1}^{n}(y_i - ct)$$

$$\Longleftrightarrow$$

$$\prod_{i=1}^{n}(x_i - dt) = d^n \cdot \prod_{i=1}^{n}(y_i/c - t)$$
$$\iff$$
$$\prod_{i=1}^{n}(x_i/d - t) = \prod_{i=1}^{n}(y_i/c - t).$$
$$\iff$$
$$\prod_{i=1}^{n}(t - x_i/d) = \prod_{i=1}^{n}(t - y_i/c).$$

Denote by P (resp. Q) the left (resp. right) hand of the last equation, that is $P \stackrel{\triangle}{=} \prod_{i=1}^{n}(t - x_i/d)$ and $Q \stackrel{\triangle}{=} \prod_{i=1}^{n}(t - y_i/c)$. P and Q can be seen as polynomials $P[t]$ and $Q[t]$ over $\mathbb{Z}_q$ in the formal variable t. If $P[t]$ and $Q[t]$ are identical polynomials (the lists of the coefficients are identical), then the statement (i.e., that there exists a permutation γ over $[n]$ such that for any $i \in [n], Y_i^d = X_{\gamma(i)}^c$), is true (the viceversa holds as well). Indeed, suppose $P[t], Q[t]$ to be identical polynomials (the list of coefficients of $P[t]$ is equal to the list of coefficients of $Q[t]$). By definition of $P[t]$ and $Q[t]$, and from the fact that a polynomial of degree n, as $P[t]$ and $Q[t]$ are, has at most n roots, it follows that there exists a permutation γ over $[n]$ such that for every $i \in [n], x_i/d = y_{\gamma(i)}/c \iff x_i \cdot c = y_{\gamma(i)} \cdot d \iff X_i^c = Y_{\gamma(i)}^d$, as it was to show. Hence, if the statement does not hold, the two polynomials have to be different.

The soundness error is the probability, over the random choices of $t \in \mathbb{Z}_q$ of the verifier, that the proof is accepted but the statement does not hold. Let us analyze the soundness error under the hypothesis that the NIZK proof has perfect soundness. This is not the case but it simplifies the analysis and we leave it as exercise to remove this simplification.

Therefore, the soundness error is equal to the probability that the last equation holds but the two polynomials $P[t]$ and $Q[t]$ are different. The probability that the last equation holds is equal to the probability, over a random $r \in \mathbb{Z}_q$, that $P[t](r) = Q[t](r)$ (that is, that the evaluation of $P[t]$ at the point r equals the evaluation of $Q[t]$ at the point r).

As a consequence, the probability that the last equation holds is equal to the probability, over the choices of $r \in \mathbb{Z}_q$, that r is a zero of the polynomial $(P - Q)[t]$. Since $P[t]$ and $Q[t]$ are monic polynomials of degree n and they are different, the polynomial $(P - Q)[t]$ has at most $n - 1$ roots, so the soundness error is $\leq (n - 1)/q$.

- HVZK. It is easy to see that the HVZK property holds and we leave it as exercise.

Remark 1 *The above protocol* $\mathsf{IP}_{\mathsf{Shffl}}$ *is a public-coin protocol and so can be made non-interactive using the FS transform, cf. Exercise 11*

Remark 2 *Observe that the the protocol* $\mathsf{IP}_{\mathsf{Shffl}}$ *can be also applied to* pairs *of group elements. Indeed, the protocol "freezes" the permutation. This makes it easy to see how to extend the previous section to shuffles of n tuples of elements in the group generated by g, and in particular to tuples of ElGamal pairs. We call the resulting protocol* $\mathsf{IP}_{\mathsf{ShfflPairs}}$.

From the n-shuffle problem to verifiable shuffles of ElGamal pairs The protocol $\mathsf{IP}_{\mathsf{Shffl}}$ can be also applied to *pairs* of group elements, as shown in [`http://courses.csail.mit.edu/6.897/spring04/Neff-2004-04-21-ElGamalShuffles.pdf`]. Moreover, therein it is shown how to remove the limitation of assuming the prove to know the discrete logs of the group elements in the statement.

This makes it easy to see how to extend the previous verifiable shuffle protocol of group elements to shuffles of n tuples of elements in the group generated by g, and in particular to tuples of ElGamal pairs. We call the resulting protocol $\mathsf{IP}_{\mathsf{ShfflPairs}}$.

We now show how $\mathsf{IP}_{\mathsf{ShfflPairs}}$ can be usedo to construct a verifiable shuffle of n ElGamal ciphertexts that suffices for our applications to e-voting. This is a verifiable shuffle from malleable encryption as described before. Recall that a ballot of a voter consists of an ElGamal ciphertext $\mathsf{CT} \triangleq (g^r, h^r \cdot m)$ (along with a signature of it that we overskip for simplicity) and we assume the underlying group has prime order q .

In the following, we present again shuffles from malleable encryption, and in particular from ElGamal, but adding the verifiability property. We assume the mixers to be the same authorities that share the secret-keys corresponding to the general public-key of the threshold ElGamal scheme.

At the beginning the list L_1 is filled with the ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ of the n voters. For $i = 1, \ldots, d$, the mixer M_i (that is also the d-th authority) performs the following operations:

- c_i is chosen randomly and independently from $\mathbb{Z}_q$ and commitment $C = g^{c_i}$ is published.
- First, all ciphertexts in the list L_i are exponentiated to the same power c to compute the ciphertexts $\mathsf{CT}'_1, \ldots, \mathsf{CT}'_n$.
- Then, choose a random permutation γ_i from $[n]$ to $[n]$ and computes the list $L'_i = (\mathsf{CT}'_{\gamma_i(1)}, \ldots, \mathsf{CT}'_{\gamma_i(n)})$. The list L'_i is published on the PBB.
- A verifiable shuffle with commitment $C = g^{c_i}$ and $D = g$ is done on the resulting ballots (i.e., the original list L_i and the new list L'_i) and a NIZK proof of correctness of this operation computed using the non-interactive version of $\mathsf{IP}_{\mathsf{ShfflPairs}}$ is published on the PBB (cf. Remark 2).
- The mixer (one of the authorities) decrypts the ciphertexts in L'_i with its share of the secret-key to output a new list L''_i of partially decrypted ciphertexts that is published on the PBB. L_{i+1} is set to L''_i. The new list L_{i+1} will be the input to the next mixer.

At this point, the messages that are decrypted are the f-th power of the original messages, where $f = \prod_{i \in [d]} c_i$. This is easily fixed by asking each authority in turn to raise the message to the $1/f$-th power. A proof of correctness of this operation can be done by each authorithy by publishing a proof that a tuple of the following form is DH: (g^c, m^c, g, m).

It is easy to see that the previous sketched protocol can be used for our e-voting application outlined at the beginning of the section. In particular, the final sequences of messages decrypted by the last mixer (i.e., last authority) will be a permutation of the voters's preferences and so the tally of the election can be computed. The universal verifiability of the NIZK proofs guarantees that all operations are performed correctly and so the tally corresponds to the actual voters' preferences.

7.3 Anonymous Voting

In all previous e-voting schemes we showed, one may not know what some voter voted for but may know whether any voter voted or not. Indeed, the digital signatures allow to check whether a given voter casted a vote or not. We say that an e-voting scheme satisfies *anonymity* if no attacker given the transcript of the election (all the strings published on the PBB) can know whether a given voter casted a ballot or not. The solution we sketch in this section allows to guarantee privacy, anonymity, eligibility and verifiability at the same time. We assume that initially all the public-keys are of the form $(g, h_i), h_i = g^{s_i}, s_i \in \mathbb{Z}_q, i \in [n]$ as it is the case for public-keys of the Schnorr's digital signature. Observe that all the keys use the same base g. For simplicity we assume that the voters vote in turn from the first to the last, that is V_1 votes then V_2 and so on until voter V_n. From the description of the protocol, it is easy to see that the protocol allows any voter to vote in any order.

With this simplification, our anonymous e-voting protocol proceeds as follows. Initially, $C_0 = g$ and list $L_1 = (h_1, \ldots, h_n)$. For $i = 1, \ldots, n$, voter V_i performs the following operations:

- Voter V_i is presented with a base C_{i-1} and the list $L_i = (h_j), j \in [n]$.
- Choose random $c_i \leftarrow \mathbb{Z}_q$, and set $C_i = C_{i-1}^{c_i}$.
- Compute the values $h'_j = h_j^{c_i}, j \in [n]$ and set $L_{i+1} = (h'_j), j \in [n]$ and publishes L_{i+1} and C_i on the PBB, The list L_{i+1} is the input to the next voter.
- Voter V_i executes a verifiable n-shuffle with commitments C_i and $D = g$ and tuples L_i and L_{i+1}.
- Voter signs her preference v_i using public-key (C_{i-1}, C_i) and secret-key c_i to obtain a signature σ_i. Both v_i and σ_i are published on the PBB and we assume that it is visible that these strings are computed by the same voter who published C_i. Observe that the signature σ_i can be verified using as

PK the pair (C_i, h'_i) and this is possible because the base g in the original Schnorr's signature is changed to C_i.

- The value h'_i is removed from the list L_{i+1}, that is the voter V_i will be denied to vote again.

Exercise 12 *Analyze the correctness and the security of the previous e-voting protocol. In particular, demonstrate that eligibility, privacy, anonymity and verifiability hold.*

8 TODO

1. definition of interactive machines with Step function.
2. put all occur
3. cryptographic shuffles rences of interactive machines M (or M_0, M_1 bla bla) with M.
4. yao's protocol
5. ..
6. ZK for NP hard language
7. NIZK in CRS model for any NP hard language of Groth Ostrovsky and Sahai and the corresponding ZAP
8. Amplifying probability of probabilistic attacks
9. BLS signatures (uses pairing)
10. Linear Encryption
11. SNARK (uses pairing)