Introduction to Cryptographic Protocols

Claude Crépeau and Simon Pierre Desrosiers

March 29, 2012

Contents

0.1 Notation

1 :	$x\|\|y$	String x is concatenated with string y.
2 :	$\lambda(C)$	Distribution of a given instance or random variable C.
3 :	$D(X,Y)$	Statistical distance between the distributions of the two random variables X and Y.
4 :	$=_0$	Equality between two Random Variables: $X =_0 Y$ is equivalent to $D(P_X, P_Y)$=0, where X and Y are random variables.
5 :	$=_\epsilon$	Epsilon distance between two Random Variables : $X{=_\epsilon}Y$ imples $D(P_X, P_Y) \leqslant \epsilon$, for random variables X and Y.
6 :	$=_{\text{Stat}}$	Statistical indistinguishability between two Random Variables: see Definition 8.
7 :	$=_{\text{Comp}}$	Computational indistinguishability between two Random Variables: see Definition 9.
8 :	$\leftrightharpoons$	Protocol proving equality between two bit commitments: $\boxed{b_0} \leftrightharpoons \boxed{b_1}$, the commitment protocol is implied by the flavor of the commitments used.
9 :	$\leftrightarrow$	When commitment $\boxed{a}$ is unveiled, it is accepted by the receiver as a valid commitment to the value b : $\boxed{a} \leftrightarrow b$.
10 :	$U(\cdot)$	The operator U is a shorthand notation representing all the info necessary to unveil a given value: $U(\boxed{a})$ is all the information necessary such that the receiver can verify that $\boxed{a} \leftrightarrow a$.
11 :	$\mathbb{U}_n$	Uniform random variable over a set of n elements or 2^n elements, the meaning will be clear from the context.
12 :	$\Pr[A(x) = y]$	Probability that probabilistic algorithm A outputs y given x.
13 :	$\Pr_x[A(x) = y]$	Average probability, over all x that probabilistic algorithm A outputs y given x : $\Pr_x[A(x) = y] = \sum_x p_x \Pr[A(x) = y]$.

0.2 Useful Formulae

1 : **Statistical Distance** Let X and Y be two random variables over the same set $\{x\}_x$, then $D(X,Y) = \frac{1}{2}\sum_x |P_X(x) - P_Y(x)|$.

2 : **Chernoff Upper Tail bound** $\Pr[X > (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$ for $\delta > 0$

3 : **Chernoff Lower Tail bound** $\Pr[X < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^\mu$ for $\delta > 0$

4 : **Bound on** e For all x we have that $1 - x \leqslant e^{-x}$.

Chapter 1

Introduction

In 1982, Goldwasser and Micali had recently introduced the notion of semantic security and established the equivalence to computational indistinguishability. Moreover, Probabilistic Encryption based on Quadratic Residuosity was published in the same paper, and proven semantically secure. At Eurocrypt '84, Silvio Micali presented a joint result with M. Fischer and C. Rackoff to the effect that Rabin's Oblivious Transfer can be secured by proving the knowledge of a square root of a quadratic residue without disclosing it !

Unfortunately, the related paper did not formalize the notions involved very rigorously and more importantly, did not appear in the conference proceedings ! However, it is totally clear that this work involved the very first Zero-Knowledge Proof of Knowledge, notions that will be introduced in the next sections. These notions were formalized by Goldwasser, Micali and Rackoff. They had a hard time getting reviewers to recognize the importance of this contribution !!! In retrospect, this is one of the most important contributions of that time period and a set of notions absolutely central to this book.

1.1 Proofs of knowledge vs proofs of membership

The seminal paper of GMR introduced two major concepts, that of an Interactive Proof and that of Zero-Knowledge. While the former may be more related to Complexity Theory, the later is clearly motivated by Cryptography. It seems quite likely that both notions were actually motivated by cryptography, but in the process of formalizing these notions, Interactive Proof probably shifted from the notion of *Proof of Knowledge* to the notion of *Proof of Membership.*

As an example, consider the statement, "This number n is such that I know its two or more prime factors". If I give you, two primes p and q such that $n = p * q$, you may check that they are indeed primes (using your favorite efficient primality test) and check that indeed $n = p * q$ (using your favorite multiplication algorithm). Because I gave you p and q, you are convinced "that I know its two or more prime factors". This is a proof of Knowledge. You discover that I know something. On the contrary, consider the statement, "This number n is the product of two or more prime factors". This statement can be verified by running your favorite test of composition. As long as n is not prime it "is the product of two or more prime factors". This is a proof of membership to the set (or language) of integers that are "the product of two or more prime factors"

As a second example, consider an valid RSA public-key (n, e). Let c be a ciphertext under this public-key. The (membership) statement "There exists a plaintext m such that $c = m^e \pmod{n}$."

is absolutely trivial because every c has this property !!! However, the (knowledge) statement "I know a plaintext m such that $c = m^e \pmod n$)." is considerably more interesting...

The notion of proof of knowledge, although it was used in [FMR84], took more time to be formalized [??] than that of proof of membership [GMR89]. We will see later that Proofs of Knowledge are central to many of our constructions.

1.2 Interactive Proofs vs Arthur-Merlin Games

The notion of interactive proofs, formalized by [GMR89] states that a computationally unbounded prover may (interactively) prove membership of an element x to a set L by speaking with a probabilistic polynomial-time (in the size of x) bounded verifier using several back-and-forth messages of polynomial length in total. What matters at the end of the conversation is (completeness:) that the verifier accepts with probability (nearly) one when $x \in L$, and (soundness:) that whatever the prover does, the verifiers accepts with probability (nearly) zero when $x \notin L$.

Because we can amplify the probabilities using a majority vote over many runs, we formally define the completeness and soundness conditions with respect to arbitrary probabilities of two thirds: for all $x \in L$, the probability that the verifier accepts is at least $2/3$, and for all $x \notin L$, the probability that the verifier rejects is at least $2/3$. The overall "polynomial length in total" bound yields to a natural complexity class called IP (for Interactive Proofs).

Independently and concurrently to Goldwasser, Micali and Rackoff, another team composed of László Babai and Shlomo Moran introduced the very similar notion of Arthur-Merlin Games. In the language of Interactive Proofs, we can express this alternate notion as a restriction of the former: a computationally unbounded prover, Merlin, may (interactively) prove membership of an element x to a set L by speaking with a probabilistic polynomial-time bounded verifier, Arthur, using several back-and-forth messages of polynomial length in total, where Arthur's messages are restricted to random coin tosses. The completeness and soundness conditions are defined exactly the same way as for Interactive Proofs. Another way of comparing the two notions is by observing that in the latter, the verifier's coin tosses are public, while in the former, the verifier's coin tosses are private. This is indeed the only difference between the two notions.

The Arthur-Merlin games of at most k alternations with Arthur speaking first AM[k], and with Merlin speaking first MA[k] are natural generalizations of NP=AM[0] and BPP=MA[0]. If we allow the number of alternation to be polynomial in the length of the input string x then we define AM[POLY]=MA[POLY].

A natural observation is that if we restrict the completeness probability to be exactly one and the completeness probability to be exactly zero then we redefine the so-called polynomial-time hierarchy. For the oracle definition of the polynomial hierarchy, define $\Delta_0^P := \Sigma_0^P := \Pi_0^P := P$. Then for $i \geq 0$ define

$$\Delta_{i+1}^P := P^{\Sigma_i^P}$$

$$\Sigma_{i+1}^P := NP^{\Sigma_i^P}$$

$$\Pi_{i+1}^P := coNP^{\Sigma_i^P}$$

where A^B is the set of decision problems solvable by a Turing Machine in class A augmented by an oracle for some complete problem in class B. For example, $\Delta_1^P := P^P = NP$, $\Sigma_1^P := NP^P = coNP$, and $\Pi_1^P := coNP^P = P^{NP}$ is the class of problems solvable in polynomial time with an oracle for some NP-complete problem. If we compare the two hierarchies, $AM[k]$ is analogous to Δ_k^P and $MA[k]$ is analogous to Σ_k^P.

Within the current knowledge of complexity theory, it is widely believed that all the classes in the polynomial-time hierarchy are strictly different from each other. It is a well known result that the least class containing all those $PH = \bigcup_{i\geq 0} \Delta_i = \bigcup_{i\geq 0} \Sigma_i = \bigcup_{i\geq 0} \Pi_i \subseteq PSPACE$. It is widely believed that on the contrary, $PSPACE \nsubseteq PH$.

It turns out, the Interactive Proofs and Arthur-Merlin Games hierarchies are full of surprises:

- Babai and Moran have demonstrated that for $k > 2, AM[2] = MA[k] = AM[K]$; in other words, the Arthur-Merlin Hierarchy collapses to its second level.
- Goldwasser and Sipser have demonstrated that for $k > 2, IP[2k] \subseteq AM[k]$; in other words, Interactive Proofs are not significantly more powerful than Arthur-Merlin Games. In particular, IP=IP[POLY]=AM[POLY].
- Shamir has demonstrated that $IP = PSPACE$; in other words, if you allow a polynomial number of alternations, whether your soundness and completeness are statistical or perfect, and whether public or private coins are used, makes absolutely no difference !

1.3 Zero-knowledge

1.4 Arguments

1.5 Bit Commitments

A commitment scheme is a two-phase cryptographic protocol between two parties, a sender and a receiver, satisfying the following constraints. At the end of the first phase (named Commit) the sender is committed to a specific value (often a single bit) that he cannot change later on (Commitments are binding) and the receiver should have no information about the committed value, other than what he already knew before the protocol (Commitments are concealing). In the second phase (named Unveil), the sender sends extra information to the receiver that allows him to determine the value that was concealed by the commitment. Commitments are important components of zero-knowledge protocols [4, 16], and other more general two-party cryptographic protocols [19]. A natural intuitive implementation of a commitment is performed using an envelope (see Figure 1). Some information written on a piece of paper may be committed to by sealing it inside an envelope. The value inside the sealed envelope cannot be guessed (envelopes are concealing) without modifying the envelope (opening it) nor the content can be modified (envelopes are binding). Unveiling the content of the envelope is achieved by opening it and extracting the piece of paper inside (see Figure 2). The terminology of commitments, influenced by the legal vocabulary, first appeared in the contract signing protocols of Even [14], although it seems fair to attribute the concept to Blum [3] who implicitly uses it for coin flipping around the same time. In his Crypto 81 paper, Even refers to Blum's contribution saying: "In the summer of 1980, in a conversation, M. Blum suggested the use of randomization for such protocols". Apparently, Blum introduced the idea of using random hard problems to commit to something (coin, contract, etc.). However, one can also argue that the earlier work of Shamir et al. [26] on mental poker implicitly used commitments as well, since in order to generate a fair deal of cards, Alice encrypts the card names under her own encryption key, which is the basic idea for implementing commitments. The term "blob" is also used as an alternative to commitment by certain authors [4,8,20]. The former mostly emphasizes the concealing property, whereas the latter refers mainly to the binding property. Under computational assumptions, commitments come in two dual flavors: binding but computationally concealing commitments and concealing but computationally binding commitments. Commitments of both types may be achieved from any one-way function [18,24,25,17].

1.6 Rudich

1.7 Oblivious Transfer

1.8 Two-party Computations

1.9 Multi-prover Interactive Proofs

Chapter 2

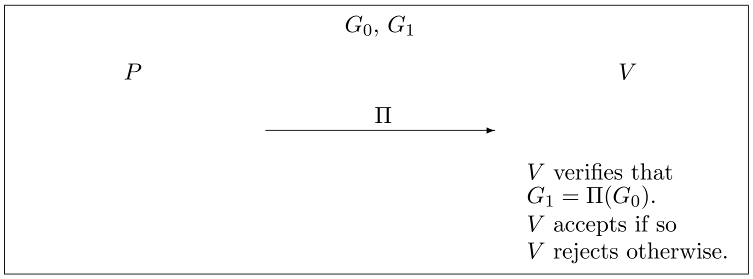
Interactive Proofs

An **Interactive Proof** is a protocol between two parties [1] a prover P and a veryfier V where P proves to V some assertion: that the string x belongs to some language L.

Take the problem of Graph-Isomorphism (G-ISO). Let $G_0 = \langle V, E_0 \rangle$ and $G_1 = \langle V, E_1 \rangle$ be two undirected graphs, where V is a set of vertices and E_i is a set of edges between the vertices of V. Furthermore $|E_0| = |E_1|$.

We say two graphs are isomorphic if and only if there exists a permutation Π of V such that for all vertices u and v in V, the edge (u, v) belongs to E_0 if and only if the edge $(\Pi(u), \Pi(v))$ belongs to E_1. We shall then write $G_0 \approx G_1$ or $G_1 = \Pi(G_0)$.

Let P and V be two Turing machines, then figure 2.1 is a protocol that lets the prover P prove to the verifier V that G_0 and G_1 are isomorphic.



Protocol 2.1: An IP protocol for Graph Isomorphism

Definition 1 (Interactive Proof) *A pair of Turing machines P and V, where machine P has no time or space limitation and machine V is probabilistic-polynomial time (PPT), constitute an IP system for the language $\mathcal{L}$ if the interaction between the two machines has the two following properties:*

***Completeness**: for all $x \in \mathcal{L}$*

$$\Pr[V \textit{ accepts } x \textit{ after interacting with } P] \geqslant \frac{2}{3} \tag{2.1}$$

***Soundness**: for all $x \notin \mathcal{L}$ and all P'*

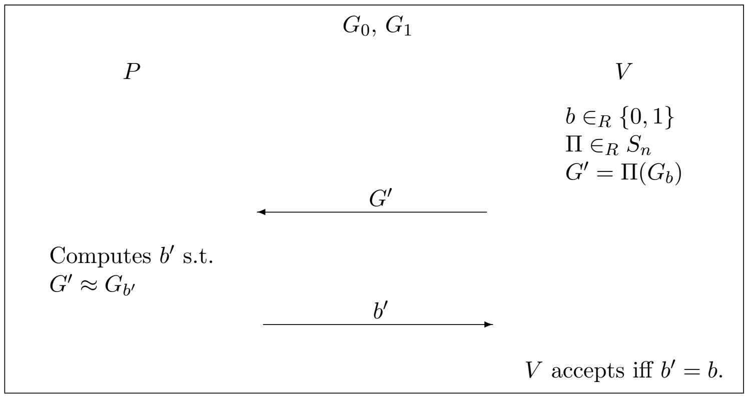
$$\Pr[V \textit{ accepts } x \textit{ after interacting with } P'] \leqslant \frac{1}{3}. \tag{2.2}$$

[1]Normally considered to be Turing Machines

A protocol is *complete* when the prover P can convince with high probability a verifier V of the veracity of the assertion $x \in \mathcal{L}$ if in fact $x \in \mathcal{L}$. In other words, the protocol is actually useful at proving to V that $x \in \mathcal{L}$. On the other hand, a protocol is *sound* when it ensures that the prover P cannot abuse V's gullibility. That is, if x is not in $\mathcal{L}$, then with high probability, if V follows the protocol, V will not believe the prover, whatever the prover does.

The set of languages for which there exists an Interactive Proof will be called **IP**. Obviously, G-ISO is in IP. The language G-ISO being in NP, there exists a witness (the permutation Π) that P can give to V and that V can verify all by himself. The same reasoning holds for all languages in NP, hence NP $\subseteq$ IP.

The problem of Graph-NonIsomorphism (GNI) is a more interesting example of an IP system as it is not known to be in NP (it is believed that no witness can be provided to the verifier by the prover). Two graphs, G_0 and G_1 are said to be non-isomorphic if no permutation exists such that $G_1 = \Pi(G_0)$. Figure 2.2 shows an IP protocol for Graph-nonisomophism. In that Protocol, if the two graphs G_0 and G_1 are isomorphic, then the prover P will not be able to guess b every time, hence, if V really chooses b randomly, then the probability that $b' = b$ is only one half.



Protocol 2.2: An IP protocol for Graph non-Isomorphism

On the other hand if the two graphs are nonisomorphic, it is always possible for a powerful prover P to find a unique b' for which there exists a permutation such that $G' = \Pi'(G_{b'})$, hence P can always win the game.

Hence

— Completeness: if $G_0 \not\approx G_1$ then $\Pr[(P,V)(G_0,G_1) = 1] = 1$

— Soundness: if $G_0 \approx G_1$ then $\forall P', \Pr[(P',V)(G_0,G_1) = 1] \leqslant \frac{1}{2}$.

Where $(P,V)(x) = 1$ means that after interacting with P on common input x, V accepts. As it is, this protocol does not satisfy definition 1, but this is only a technical issue. If P and V repeat the protocol twice (and V accept if and only if he accepts in both runs), then this is an IP protocol. By repeating the protocol, we mean that V's random bit and random permutation are chosen anew in this second round and independently from the first round. Hence, the two rounds being independent, soundness drops to one quarter (below one third) and therefore satisfies the definition.

By repeating several times this two-step protocol, the verifier can be convinced up to an exponentially low error probability of the validity of the proof. Note that GNI is not known to belong to NP but is in Co-NP and yet GNI belongs to IP. In fact, it turns out as proven by Adi Shamir that IP=PSPACE.

In general, we can always amplify the probability that V accepts or rejects by repeating a protocol. In the case where the completeness is smaller than 1, the verifier will run a given protocol which belongs to IP k times recording each time whether he accepted or rejected the prover's claim. At the end of the k rounds, the verifier takes a majority vote : he accepts the prover's claim, if at least $\lfloor k/2 \rfloor + 1$ time, he accepted the IP proof and rejects otherwise. Now let's consider the case where x does not belong to the language but the prover is trying to prove otherwise. What is the probability of error p_e, that is the probability that V accepts x as belonging to $\mathcal{L}$?

The verifier will accept if and only if he accpeted in at least $\lfloor k/2 \rfloor + 1$ rounds. Hence, since the soundness probability is no larger than one third, and that all k rounds are independent, the expected value of the number of times V accepted x over k runs is at most $k/3$. To err, it must be that the number of times where V accepts is deviating by at least $k/6+1$ from its expected value $k/3$. But by the Chernoff bound (Formula 3), the probability that this happens is exponentially small for sufficiently large k.

If $x \in \mathcal{L}$, then if $\Pr[(P,V)(x)=1] \geqslant \frac{2}{3}$ we can amplify to $\Pr[(P,V)(x)=1] \geqslant 1-\varepsilon$ where $\varepsilon = \frac{1}{poly(|x|)}$ or more precisely $\varepsilon = \frac{1}{exp(|x|)}$.

Definition 2 (Negligible function) *A function $\mu : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for every positive polynomial $p(\cdot)$ there exists an n_0 such that for all $n > n_0$ we have*

$$\mu(n) < \frac{1}{p(n)}.$$

Interactive Proofs where $\Pr[V \text{accepts } x \text{ after interacting with } P] = 1$ for all $x \in \mathcal{L}$ can be amplified faster by taking advantage of this special property. If we repeat k times and accept if and only if all executions accept then the completeness probability remains one while the soundness probability drops to p_s^k if p_s was the soundness probability of a single run. Notice that we have used a distinct accpeting criteria. This technique can be used whenever $p_s < 1$, not only for $p_s < \frac{1}{3}$.

2.1 Proof of Knowledge

A proof of knowledge is a variation on IP protocols. In this case, the goal is to convince a verifier that the prover *knows* something. For example, the prover might want to convince the verifier that he *knows* the two prime factors, p and q, of a given RSA number $n = pq$. The difference with IP protocols is that we need to formally define a notion of *knowledge* to exclude the possibility that the prover proves that n has exactly two prime factors without knowing them.

Consider a relation $\mathcal{R}$ constituted of pairs (x, w), where w is a witness. For example, the relation for RSA numbers is the set $\{(n,(p,q)) | n = pq, p \text{ and } q \text{ are prime}\}$; or a pair which constitutes a witness to the RSA-Number language. Proving that n is composite does not require knowledge of p and q because that's exactly the outcome of Rabin's primality test. Proving that n has exactly two prime factors seems much harder to do without knowledge of the explicit factors.

Another good example is $((G_0, G_1), \pi)$ where (G_0, G_1) forms the language of pairs of isomorphic graphs, and π is the permutation such that $G_1 = \pi(G_0)$. In this last case, a proof of knowledge is kind of an IP protocol such that at the end, not only is the verifier convinced that $G_0 \approx G_1$, but the verifier is also convinced that indeed the prover knows the permutation π.[2]

The language $\mathcal{L}_\mathcal{R}$ associated with $\mathcal{R}$ is the set of all x such that there exists a witness for which $\mathcal{R}(x, w) = 1$: $\mathcal{L}_\mathcal{R} \triangleq \{x | \exists w \text{ s.t. } \mathcal{R}(x, w) = 1\}$. An *explicit* proof of knowledge, would be, for instance, the interaction where P provides V with the witness w (as in protocol 2.1). However, the notion of proof of knowledge is a lot more subtle. If w appears in the conversation between P and V in an implicit form, say $\bar{w}$ for instance, it is still a proof of knowledge. That's because there exists an efficient algorithm that computes w from $\bar{w}$ (by flipping all the bits), and more precisely w can efficiently be computed from the conversation between P and V. We shall call this algorithm the **Knowledge Extractor**. To give a very general notion of *implicit knowledge* we push the definition of Knowledge Extractor even further: we say that an efficient algorithm K_P is a knowledge extractor for a relation $\mathcal{R}$ if given $x \in \mathcal{L}_\mathcal{R}$, $\mathcal{R}(x, K_{P(w)}(x)) = 1$ with non-negligeable probablity. When no such witness exists, we don't care what $K_{P(w)}$ outputs. So here is the formal definition:

Definition 3 (Proof of Knowledge) *A pair of (Probabilistic Polynomial-time) Turing machines, P and V, constitute an interactive proof of knowledge for a relation $\mathcal{R}$ if the following two conditions hold*

***Completeness**: for all $(x, w) \in \mathcal{R}$*

$$\Pr[(P(w), V)(x) = 1] \geqslant 1 - \nu(|x|) \tag{2.3}$$

***Soundness**: for all prover P' there exists an efficient [3] $K_{P'}$ such that for all x and all w we have*

$$\Pr[\mathcal{R}(x, K_{P'(w)}(x)) = 1] \geqslant \Pr[(P'(w), V)(x) = 1] - \kappa(|x|), \tag{2.4}$$

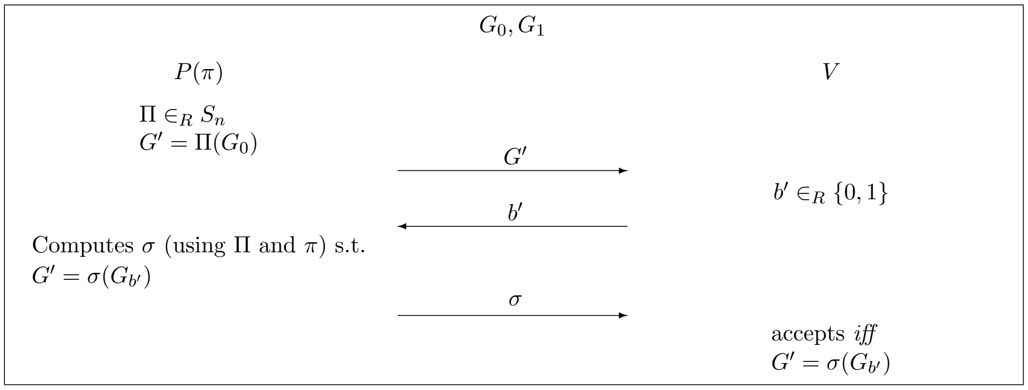
where κ and ν are negligeable error-functions.

Notice that when $x \notin \mathcal{L}_\mathcal{R}$ we have that $\Pr[\mathcal{R}(x, K_{P'(w)}(x)) = 1] = 0$ and therefore $\kappa(|x|) \geqslant \Pr[(P'(w), V)(x) = 1]$ which is a strong soundness condition of IP protocols. Therefore, all proofs of knowledge are IP protocols but not the other way around... Intuitively, this definition states that whenever a prover P' given a candidate-witness w manages to convince a verifier that $x \in \mathcal{L}_\mathcal{R}$ then we can obtain a witness (not necessarily w) by running $K_{P'(w)}(x)$ with similar probability in an amount of time not too much bigger...

As mentioned before, protocol 2.1 is an explicit proof of knowledge of the isomorphism between G_0 and G_1. The following protocol 2.3 will be the basis for an implicit proof of knowledge for the same witness.

[2]The fact that a prover can convince a verifier that two graphs are isomorphic does not imply that the prover knows the permuatation π, even if it is hard to imagine otherwise.

[3]We allow $K_{P'(w)}$ to run for polynomially more time than $P'(w)$. One simple way of enforcing this condition is to define $K_{P'}$ as a poynomial-time algorithm that runs P' as a black-box.

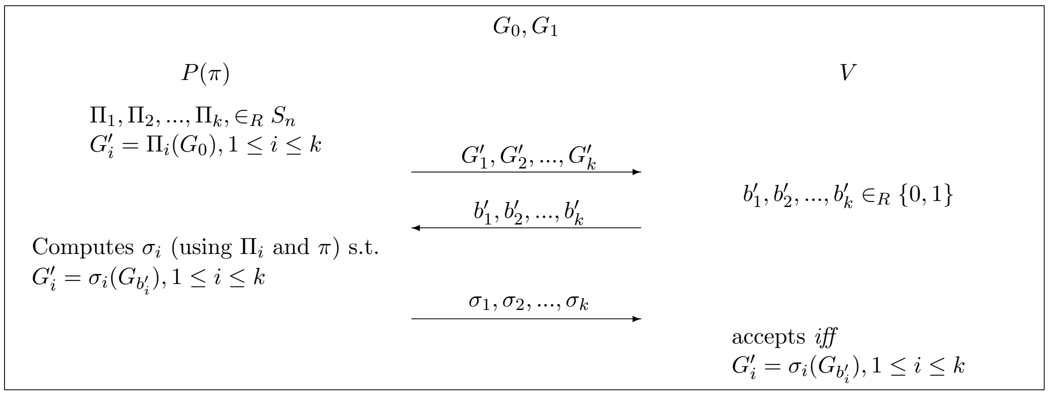


Protocol 2.3: Basic protocol for an implicit proof of knowledge of Isomorphism between two Graphs

First, check that protocol 2.3 is an IP protocol for GI. To convince yourself of this fact, check that the completeness condition is perfect: $\Pr[(P,V)(x) = 1] = 1$. Whereas the IP soundness condition is also satisfied, when G_0 and G_1 are not isomorphic, $\Pr[(P'(\pi),V)(x) = 1] \leq \frac{1}{2}$ whatever P' does (the last step of the protocol can be acomplished properly by P' for only one value of b' because G' cannot be isomorphic to two graphs that are not isomorphic to each other).

As is, the above protocol does not constitute a proof of knowledge according to our above definition. That's because the soundness success probability of a prover P' that knows no witness can be as much as $\frac{1}{2}$. This would imply that the knowledge extractor K_P should have $\Pr[\mathcal{R}(x, K_{P'}(x)) = 1] \geqslant \frac{1}{2} - \kappa(|x|) > 0$ which is impossible when $x \notin \mathcal{L}$.

Consider instead, executing k independant copies of protocol 2.3 for the same input graphs G_0, G_1 as follows:


G_0, G_1
$P(\pi)$
V
$\Pi_1, \Pi_2, ..., \Pi_k, \in_R S_n$
$G'_i = \Pi_i(G_0), 1 \leq i \leq k$
$G'_1, G'_2, ..., G'_k$
$b'_1, b'_2, ..., b'_k \in_R \{0,1\}$
$b'_1, b'_2, ..., b'_k$
Computes σ_i (using Π_i and π) s.t.
$G'_i = \sigma_i(G_{b'_i}), 1 \leq i \leq k$
$\sigma_1, \sigma_2, ..., \sigma_k$
accepts *iff*
$G'_i = \sigma_i(G_{b'_i}), 1 \leq i \leq k$

Protocol 2.4: Implicit proof of knowledge of Isomorphism between two Graphs

The resulting protocol still has completeness probability $\Pr[(P(\pi),V)(x) = 1] = 1$. Whereas the strong IP soundness condition is also satisfied, when G_0 and G_1 are not isomorphic, $\Pr[(P'(\pi),V)(x) = 1] \leq \frac{1}{2^k}$ whatever P' does. Moreover, the stronger proof of knowledge soundness condition with $\kappa(|x|) = \frac{1}{2^k}$ is also satisfied:

$$\Pr[\mathcal{R}(x, K_{P'}(x)) = 1] \geqslant \Pr[(P'(\pi),V)(x) = 1] - \kappa(|x|).$$

When $x \notin \mathcal{L}$, we get $\Pr[\mathcal{R}(x, K_{P'}(x)) = 1] = 0$ which yields $\Pr[(P'(\pi),V)(x) = 1] \leq \frac{1}{2^k}$. When $x \in \mathcal{L}$, we get $\Pr[\mathcal{R}(x, K_{P'}(x)) = 1] \geqslant \epsilon(x) - \kappa(|x|)$ where $\epsilon(x) = \Pr[(P'(\pi),V)(x) = 1]$. If P' is so dum that $\epsilon(x)$ is negligeable, then $\epsilon(x) - \kappa(|x|)$ is negligeable (or even negative!!) and therefore $\Pr[(P'(\pi), V)(x) = 1]$ may also be negligeable. In this case, there is no requirements on $K_{P'}$ to succeed at all. If however, P' is such that $\epsilon(x)$ is not negligeable then we expect that $K_{P'}$ will succeed with probability at least $\epsilon(x)$. We now present a polynomial time $K_{P'}$ that succeeds in producing a witness with probability nearly 1 under the condition that $\epsilon(x)$ is not negligeable.

The purpose of the knowledge extractor $K_{P'}$ is to obtain two executions of P' such that the graph G' submitted in both runs are identical but the choice bit b' of the extractor are distinct. Here is the knowledge extractor for the Protocol 2.4

```
1. Initialize $P'$: copy fresh random bits to the prover's random tape and fill up
   the Auxiliary-Input tape with a witness $\pi$ if any.
2. Run $P'$ until it sends $G'_1, G'_2, ..., G'_k$
3. Send random $b'_1, b'_2, ..., b'_k$ to $P'$ and wait for $\sigma_1, \sigma_2, ..., \sigma_k$
4. Store $d'_1, d'_2, ..., d'_k \leftarrow b'_1, b'_2, ..., b'_k$ and $\gamma_1, \gamma_2, ..., \gamma_k \leftarrow \sigma_1, \sigma_2, ..., \sigma_k$
5. Restart $P'$ as in step 2 and run it until it sends $G'_1, G'_2, ..., G'_k$ again
6. Send random $b'_1, b'_2, ..., b'_k$ to $P'$ and wait for $\sigma_1, \sigma_2, ..., \sigma_k$
7. Let $i$ be such that $b'_i \neq d'_i$. Compute and output $\pi = \sigma_i^{-1^{b'_i}} \gamma_i^{-1^{d'_i}}$. STOP
8. Go to step 5.
```

Extractor 2.1: Sketch of the Knowledge Extractor for Graph Isomorphism

The Extractor above is just a sketch because many subtleties have to be considered. As written, we assume that P' is always answering valid $\sigma_1, \sigma_2, ..., \sigma_k$ for arbitrary $G'_1, G'_2, ..., G'_k$ and $b'_1, b'_2, ..., b'_k$. Let p be the probability that P' actually answers valid $\sigma_1, \sigma_2, ..., \sigma_k$ at step 3. If the first time the extractor tries this Step, P' answers with invalid $\sigma_1, \sigma_2, ..., \sigma_k$ then the extractor aborts. The running time of this possibility is independent of p. If the first time the extractor tries this Step, P' answers with valid $\sigma_1, \sigma_2, ..., \sigma_k$ then the extractor should extract a withness. It will do that by finding another set of $b'_1, b'_2, ..., b'_k$ for the same $G'_1, G'_2, ..., G'_k$ that produces valid $\sigma_1, \sigma_2, ..., \sigma_k$. If p was the probability of hitting a situation that lead P' to issue valid $\sigma_1, \sigma_2, ..., \sigma_k$ then the probability that this happens again is still p. Therefore, the expected number of tries until this situation happens again is $1/p$. Therefore the expected running time to produce a witness is $p \times 1/p \times t$ where t is the time to run one test at Steps 5–6. This expected running time is again independent of p.

Nevertheless, this argument is slightly wrong: the probability p is computed as an expected value over the choices of $b'_1, b'_2, ..., b'_k$. The Extractor fails if the Prover completes the protocol consistently on the same sequence $b'_1, b'_2, ..., b'_k$. The extreme example is if the Prover solely completes on a single sequence $b'_1, b'_2, ..., b'_k$. In this case, the Extractor would not succeed because he never finds a second sequence of $b'_1, b'_2, ..., b'_k$'s. To escape this undesired situation, the extractor always runs in parallel to Step 5 a complete enumeration of all possible permutations until he finds one that maps G_0 to G_1. If by any chance, it runs long enough that this enumeration completes then the Extractor output the computer permutation. This case will arise with probability no greater than 2^{-k}, when no other sequence succeeds. All in all, this will double the expected running time of any situation where ultimately a second sequence is found, while it will force termination of any situation where no other sequence is actually found. This will only increase the expected running time of the Extractor by a factor of two as long as 2^k is larger than the number of permutations.

Problems

2.1 Formalize the argument for majority amplification using the Chernoff bound. Do this for both Soundness and completeness.

2.2 Let n be a composite number chosen by the Prover. Let y be a quadratic non-residu modulo n (with Jacobi symbol $+1$) also chosen by the Prover. Give an interactive proof for the language

$$N\text{-}QNR = \{(n,y)|n \text{ is composite and } y \text{ is a quadratic non-residu mod } n \text{ (with Jacobi symbol } +1)\}.$$

2.3 Let n be a composite number and y be a quadratic non-residu modulo n (with Jacobi symbol $+1$) chosen by the Prover. Let $z \in QR_n \bigcup yQR_n$ where we define $yQR_n = \{z \in QNR_n | z/y \in QR_n\}$. Give a proof of knowledge for the residuosity of z where the prover convinces the verifier that he knows whether z is a quadratic residue or non-residue but does not tell the verifier which it is...

2.4 Let p be a public prime number with publicly known factorization of $p-1$ and g be a public primitive element. Let (a,b,c) be a triple of elements from $\mathbb{Z}_p^*$ chosen by the Prover as $a = g^x \bmod p$, $b = g^y \bmod p$ and $c = g^{xy+z} \bmod p$ for some x, y, z. Give a proof of knowledge for the Decisional Diffie-Hellman set such that $(a,b,c) \in DDH$ if and only if there exists x, y such that $a = g^x \bmod p$, $b = g^y \bmod p$ and $c = g^{xy} \bmod p$. The prover convinces the verifier that he knows whether (a,b,c) is a valid DDH triple or not but does not tell the verifier which it is...

Hint: Give a proof of knowledge of z without disclosing it.

2.5 Let p be a public prime number with publicly known factorization of $p-1$ and g be a public primitive element. Suppose the Prover has free access to an oracle to decide the DDH. Show how to use the protocol of 2.4 in order to convince the verifier that he is able to decide DDH. Make sure that the verifier never learns whether a triple (a,b,c) is a valid DDH-triple or not, unless he already knows...

Chapter 3

Zero-Knowledge

The proofs seen in the last chapter are interesting, but not so much cryptographically as the verifier may learn everything. Once the verifier learns the witness to an NP assertion, then he can himself prove the assertion to anyone else. In this chapter we shall develop another property for protocols. We shall require that not only do they convince the verifier of the validity of the assertion but that the verifier learns *nothing else* from the interaction with the prover. After the proof, the verifier is really convinced of the validity of the assertion that the prover was in fact proving. Ideally, the verifier should learn nothing; so little that the verifier cannot even convince his verifier-friends that he indeed talked to the Prover (or anyone who knew a witness). We shall do that via a very profound technique which is known as *simulation.* The result of this definition will be a powerful new tool.

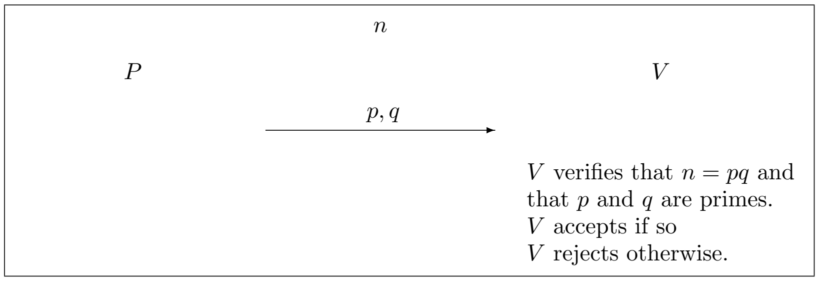
We shall simulate the interaction of V', for any V', even a potentially dishonest V', with P without having access to P or his knowledge. Let $VIEW[(P,V')(x)]$ be the random variable associated with all the possible transcripts of all the messages exchanged between P and V' and all of V''s random coins and x (basically, anything that V' knows and sees). We shall call the *transcript* the list of all messages exchanged between P and V. Clearly, the View contains more information than the transcript.

Definition 4 *An IP protocol is Zero-Knowledge (ZK) if for all verifiers V' there exits a probabilistic polynomial-time machine $S_{V'}$ such that for all $x \in \mathcal{L}$*

$$VIEW[(P,V')(x)] =_0 S_{V'}(x). \tag{3.1}$$

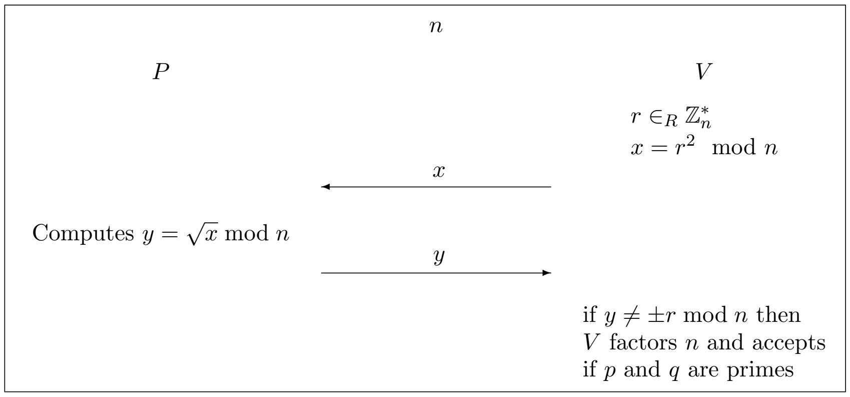
In this definition, $S_{V'}(x)$ is the random variable that represents the output of the simulator S that has access to V' (as a black-box or an oracle) and the notation $X =_0 Y$ means that the two random variables X and Y have the exact same distribution. In layman words, an IP protocol is Zero-Knowledge if there exists a PPT machine that can simulate the interaction between P and V' without knowing anything specific about x or $\mathcal{L}$ (a witness for example).

Let us look at factorization as a first (*bad*) example. Here, in Protocol 3.1, $n = pq$, or the language $\mathcal{L}$ is the set of all numbers with exactly two prime factors.



Protocol 3.1: A first attemp at Zero-knowledge

Of course, Protocol 3.1 cannot be Zero-Knowledge as no simulator that does not know p and q, which for large n could be hard to obtain, can simulate the interaction between any V' and P efficiently. But even worse, V learns the factorization of n. As we shall see, a protocol is Zero-Knowledge if the verifier learns nothing but the validity of the assertion even if the verifier does not behave honestly. This is proven by exhibiting a simulator whose outputs is distributed as the View of V'. But how can this be simulated without knowing p and q? In fact, Protocol 3.1 has the intent of giving p and q to V. In this case obviously V learns more than just the fact that n has two factors. We can try a second approach where the factors of n do not appear explicitely on the transcript, see protocol 3.2.


n
P
V
$r \in_R \mathbb{Z}_n^*$
$x = r^2 \mod n$
x
Computes $y = \sqrt{x} \bmod n$
y
if $y \neq \pm r \bmod n$ then
V factors n and accepts
if p and q are primes

Protocol 3.2: A second attemp at Zero-knowledge

But alas, even if p and q are per say not on the transcript, V still learns the factorization and no polynomial-time S should be able to simulate this protocol as extracting square roots is as hard as factoring (if P can reliably accomplish the expected task then V can use him to find p and q efficiently).

Take a look at the representation of the Verifier of Figure 3.3.

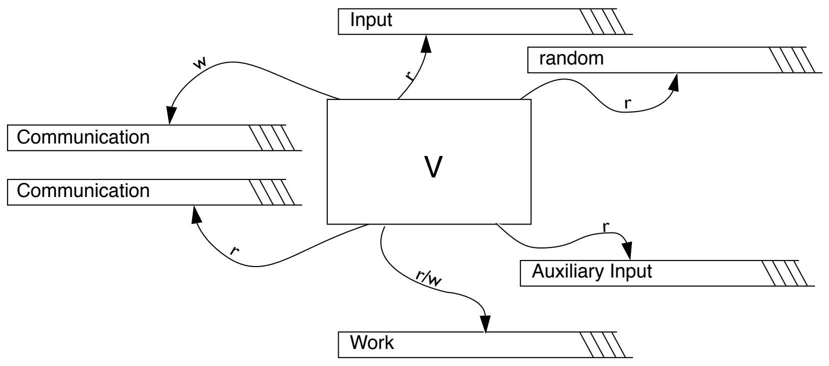


Figure 3.3: Schematization of the Verifier Turing Machine

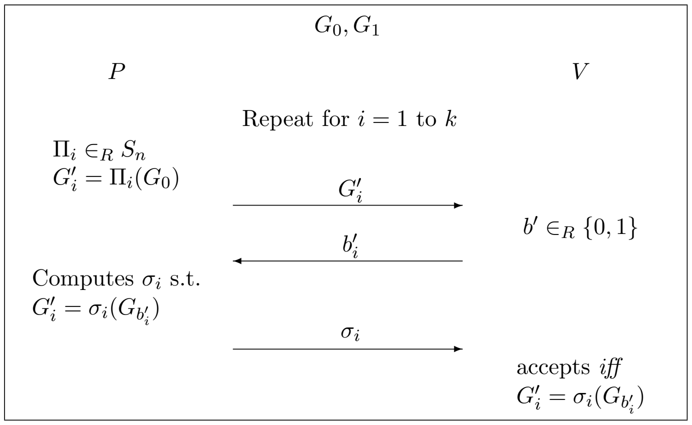
This Turing machine has 6 tapes: two of which are used to communicate with the prover, one contains the input, one is prefllled with unbiased random bits, one is a work tape and the last one contains the auxiliary input. The auxiliary input represents some prior knowledge that the verifier might posess: For example, the verifier might know that p in Protocol 3.1 is congruent to $3 \mod 4$ with probability one quarter. The definition of Zero-Knowledge should hold even if V' already knows something, that is V' should not know more after the protocol than before the protocol and whatever (computational) uncertainty he had about the world has not changed. Look at protocol 2.2 for Graph-Non-Isomorphism. Imagine that V' in this protocol knows a third graph G_2 and is trying to know to which graph, G_0 or G_1, G_2 is isomorphic. Then, by cheating and sending graph G_2 instead of choosing between G_0 and G_1 randomly and permuting it, then V' could learn something it is not suppose to learn that is $G_2 \approx G_b$ for $b \in \{0, 1\}$. Note that we never claimed that Protocol 2.2 was safe in that respect.

We can revisit the definition of Zero-Knowledge to protect the prover against such bad behaviors from V'.

Definition 5 *An IP protocol is Zero-Knowledge (ZK) if for all verifiers V' there exits a probabilistic polynomial-time machine $S_{V'}$ such that for every $x \in \mathcal{L}$ and for all auxiliary input $\aleph$ we have*

$$VIEW[(P, V'(\aleph))(x)] =_0 S_{V'(\aleph)}(x). \tag{3.2}$$

Let us revisit Protocol 2.3 that we used in the context of Proof of Knowledge previously:



Protocol 3.4: A successful attempt at Zero-Knowledge: Graph-Isomorphism

Note that in this protocol, σ is easy to compute for the prover: if $b' = 0$, then $\sigma = \Pi$ and if $b' = 1$ then $\sigma = \Pi \circ \rho$ where ρ is a permutation such that $G_1 = \rho(G_0)$.

This protocol obiously belongs to IP if it is repeated k times:

— Completeness: If $G_0 \approx G_1$, then $\Pr[(P, V)(x) = 1] = 1$,

— Soundness: If $G_0 \not\approx G_1$, then $\forall P' \Pr[(P', V)(x) = 1] \leqslant \frac{1}{2^k}$.

But contrary to what we did in the proof of knowledge of Protocol 2.4, to obtain Zero-Knowledge the repetitions in Protocol 3.4 are *sequential* instead of in parallel.

To prove that this protocol is Zero-Knowledge, we only have to present a simulator, see Figure 3.5.

1. Copy fresh random bits to the verifier's random tape and fill up the Auxiliary-Input tape.
2. Pick a bit $c \in_R \{0,1\}$ and permutation $\rho \in_R S_n$.
3. Compute $G' = \rho(G_c)$
4. Run the verifier V on input (G_0, G_1) and G' and wait for his reply b'.
5. (a) If $b' = c$, then send ρ and finish simulation
   (b) if $b' \neq c$, then rewind V as if nothing had happened and go back to step 2

Simulator 3.5: Simulator for Graph-Isomorphism

Here are a few comments on the simulator for Graph-Isomorphism:

1. As long as V runs in polynomial time, the simulator runs in expected polynomial time.
2. The simulator has access to all the tapes of V in read-write mode but does not have access to the internal logic of V.
3. The verifier has a reset switch to which S has access, using this button the simulator can rewind V to its initial state (right after step 1 in Simulator 3.5).

4. The distribution of G' does not depend on c.
5. Since $G_0 \approx G_1$, V' cannot know if G' was generated applying a permutation to G_0 or G_1.
6. The two distributions $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are identical.
7. The bit c never appears on the transcript.

To enlighten some of these comments, let us prove that $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are identically distributed. A View is the following tuple $(G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1), \dots, (G'_k, b'_k, \sigma_k))$, where r is the content of V''s random tape which is constituted of uniform random bits and $\aleph$, the auxiliary input, is the same in both cases:

We shall write, for $j \leqslant k$,

$$(G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1), \dots, (G'_j, b'_j, \sigma_j))^S$$

for a partial View generated by the simulator and

$$(G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1), \dots, (G'_j, b'_j, \sigma_j))^P$$

for a partial View generated by a real interaction between P and V'. Note that these Views are really random variables. What we shall prove is that for any j the distribution of the real and of the simulated Views are the same.

The first thing to notice is that at any given step j, V' is really just a deterministic function of the prefix of the partial View up to that step. So before that first step we have

$$(G_0, G_1, r, \aleph)^S = (G_0, G_1, r, \aleph)^P \tag{3.3}$$

The next item on the View is G'_1. But the honest prover and the simulator create that graph according to the same distribution: that is they use the graph G_0 (or G_1 at random for the simulator) and compute a random isomorphism of it (by choosing uniformly a random permutation). But since G_0 and G_1 are isomorphic, a random permutation of either is the same. Hence

$$(G_0, G_1, r, \aleph, (G'_1))^S =_0 (G_0, G_1, r, \aleph, (G'_1))^P. \tag{3.4}$$

The next element, which is V''s challenge to G'_1 is a deterministic function of the View up to that point, as V' uses the bits in r as randomness. Hence Equation (3.4) means that

$$V'((G_0, G_1, r, \aleph, (G'_1))^P) =_0 V'((G_0, G_1, r, \aleph, (G'_1))^S),$$

hence

$$(G_0, G_1, r, \aleph, (G'_1, b'_1))^S =_0 (G_0, G_1, r, \aleph, (G'_1, b'_1))^P. \tag{3.5}$$

And finally as both the prover and the simulator chose their permutation to create G'_1 at random uniformly, and that for a fixed permutation γ, $\{\Pi \circ \gamma\}_\Pi = S_n$ the probability to see a given permutation is the same in both Views

$$(G_0, G_1, r, \aleph, (G'_1, b'_1, \rho_1))^S =_0 (G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1))^P. \tag{3.6}$$

However, the Simulator is only able to complete this part of the View when $b'_1 = c$. On the contrary, the Prover can always complete it because he knows an isomorphism between G_0 and G_1. Nevertheless, the choice of b'_1 by V' is completely independent of c since the distribution of G'_1 is the same for both values of c. This observation is absolutely crucial. This is the reason why the Simulator may fail and that doing so will not skew the distribution of the simulated View. If the Simulator "forgets" that he tried and fails (rewinds) and try again, the distribution will be exactly the same again next time... Notice that if we ran the proof in parallel, the success probability of the simulator at Step 5a would be 2^{-k} which would lead its running time to be exponential. That is why we favor runing the protocol sequentially.

This reasoning is for the first triple of the View. But the same argument will hold for all steps j larger than 1 as well (as long as the rewinding used by the simulator loops back to the most recent execution of Step 2. Once an iteration is successfully simulated, the simulator never tries to undo it.). Hence

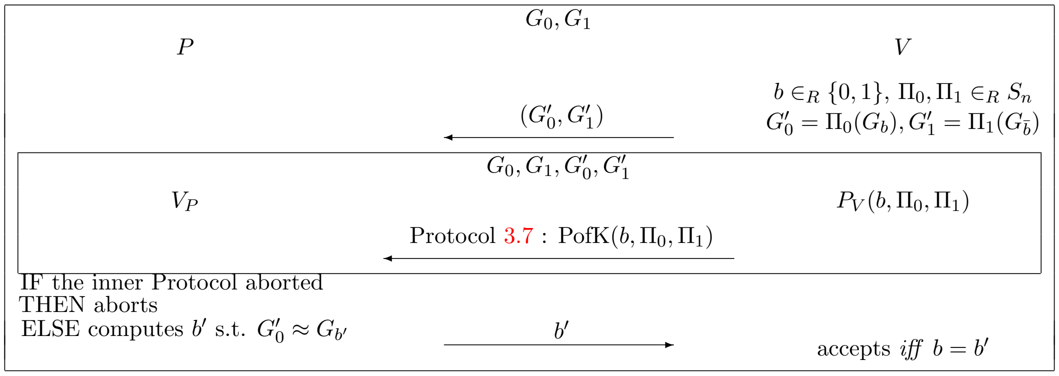
$$VIEW[(P, V'(\aleph))(x)] = S_{V'(\aleph)(x)},$$

that the simulator outputs can be the View of the verifier and this happens with the exact same probability.

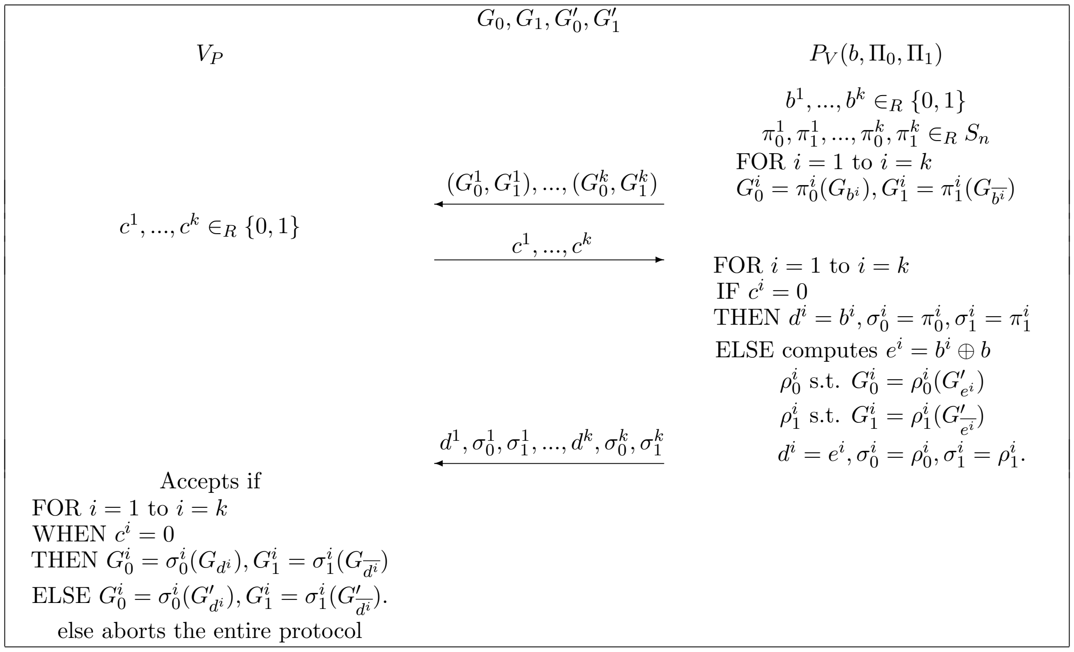
3.1 Graph Non-Isomorphism

Let us discuss a more elaborate example : a Zero-Knowledge Interactive Proof for Graph-Non-Isormophism.

Take a look back at Protocol 2.2. As we have already discussed, this protocol cannot be Zero-Knowledge as V' might not compute G' honestly. In fact, a clever V' could feed a third graph G_2 and learn whether G_2 is isomorphic to G_0 or G_1. One solution would be to force V' to compute G' honestly. Fortunately we can use a variation on the Proof of Knowledge for Graph-Isomorphism that we already know (Protocol 2.4) in order to achieve this. The following Protocol is an improvement of Protocol 2.2 by introducing an extra Step represented by the internal box.


G_0, G_1
P
V
$b \in_R \{0,1\}$, $\Pi_0, \Pi_1 \in_R S_n$
$G'_0 = \Pi_0(G_b), G'_1 = \Pi_1(G_{\bar{b}})$
(G'_0, G'_1)
G_0, G_1, G'_0, G'_1
V_P
$P_V(b, \Pi_0, \Pi_1)$
Protocol 3.7 : PofK(b, Π_0, Π_1)
IF the inner Protocol aborted
THEN aborts
ELSE computes b' s.t. $G'_0 \approx G_{b'}$
b'
accepts *iff* $b = b'$

Protocol 3.6: Zero-Knowledge proof for Graph-Non-Isomorphism



```
$G_0, G_1, G'_0, G'_1$

$V_P$                                             $P_V(b, \Pi_0, \Pi_1)$

                                                  $b^1, ..., b^k \in_R \{0,1\}$
                                                  $\pi_0^1, \pi_1^1, ..., \pi_0^k, \pi_1^k \in_R S_n$
                                                  FOR $i = 1$ to $i = k$
                  $\xleftarrow{(G_0^1, G_1^1), ..., (G_0^k, G_1^k)}$     $G_0^i = \pi_0^i(G_{b^i}), G_1^i = \pi_1^i(G_{\overline{b^i}})$
$c^1, ..., c^k \in_R \{0,1\}$
                  $\xrightarrow{c^1, ..., c^k}$
                                                  FOR $i = 1$ to $i = k$
                                                  IF $c^i = 0$
                                                  THEN $d^i = b^i, \sigma_0^i = \pi_0^i, \sigma_1^i = \pi_1^i$
                                                  ELSE computes $e^i = b^i \oplus b$
                                                      $\rho_0^i$ s.t. $G_0^i = \rho_0^i(G'_{e^i})$
                                                      $\rho_1^i$ s.t. $G_1^i = \rho_1^i(G'_{\overline{e^i}})$
                  $\xleftarrow{d^1, \sigma_0^1, \sigma_1^1, ..., d^k, \sigma_0^k, \sigma_1^k}$     $d^i = e^i, \sigma_0^i = \rho_0^i, \sigma_1^i = \rho_1^i$.
Accepts if
FOR $i = 1$ to $i = k$
WHEN $c^i = 0$
THEN $G_0^i = \sigma_0^i(G_{d^i}), G_1^i = \sigma_1^i(G_{\overline{d^i}})$
ELSE $G_0^i = \sigma_0^i(G'_{d^i}), G_1^i = \sigma_1^i(G'_{\overline{d^i}})$.
else aborts the entire protocol
```

Protocol 3.7: Proof of Knowledge of (b, Π_0, Π_1)

The Internal box is really a Proof of Knowledge going backwards : the Verifier $P_V(b, \Pi_0, \Pi_1)$ proves to the prover V_P that he knows b such that $(G'_b, G'_{\overline{b}}) = (\Pi_0(G_0), \Pi_1(G_1))$. First, let's check that the modified Protocol is still an Interactive Proof for Graph Non-Isomorphism. It should be obvious that the completeness of the outer Protocol still holds : the sub-Protocol does not change that. We have to work harder to prove that the soundness of the outer Protocol still holds. Soundness could be lost if somehow b was communicated to V_P during the sub-protocol. If we prove that everything that happens inside the internal box is statistically independent from the bit b, we demonstrate that this issue is not a problem. If that is the case, then the internal protocol carries no information about b and hence, cannot help P' cheat soundness if the two graphs are in fact isomorphic.

It is interesting to notice that we do not need to guarantee that the internal protocol carries no information about b if the two graphs are in fact non-isomorphic. Moreover, in the case where the graphs are isomorphic, we do not require the sub-protocol to be zero-knowledge, but simply carries no information about b. If P is infinitely powerful, as we often assume in Interactive Proofs, the zero-knowledge property (efficiency of a simulator) of the sub-Protocol would be totally irrelevant.

For the sake of argument, let us assume that G_0 and G_1 are isomorphic to start with. Then, for honest P_V, all graphs in the sub-Protocol will be isomorphic to each other. Hence, even for an infinitely powerful prover $V_{P'}$ it is impossible to compute b nor any of the b_i's from (G_0, G_1), (G'_0, G'_1) and all the (G_0^i, G_1^i) because they are all isomorphic to G_0 (and to G_1). Then answer to c_i is also totally independent from b as b^i, π_0^i and π_1^i are all chosen at random independently from b. The same holds for e^i, ρ_0^i and ρ_1^i, as e^i is really a one-time-pad encryption of b using b^i as a key. But since nothing was learned about the b_i's from (G_0, G_1), (G'_0, G'_1) and all the (G_0^i, G_1^i)'s, b is perfectly encrypted. Hence we conclude that when the time comes to compute b' even an infinitely P' cannot have better probability of guessing b than one half (that is if $G_0 \approx G_1$).

After checking that the modified Protocol is still an Interactive Proof, let's consider its Zero-Knowledge aspect. Here is the simulator for the Protocol 3.6

```
1. Initialize $V$:  copy fresh random bits to the verifier's random tape and fill
   up the Auxiliary-Input tape.
2. Run $V$ until it sends $(G_0^1, G_1^1)$
3. Run the Knowledge-Extractor on $P_V(b, \Pi_0, \Pi_1)$ to obtain $(b, \Pi_0, \Pi_1)$.
4. Run the full protocol as honest $P$ would until $V$ waits for $b'$ and give $b'$ as
   extracted in Step 3.
```

Simulator 3.8: Simulator for Graph Non-Isomorphism

This simulator is written with honest V in mind, but something could go wrong with this simulator if V' is cheating. What if there is no bit b such that $(G_0, G_1) \approx (G'_b, G'_{\bar{b}})$, or $P_{V'}$ does not know it. The simulator will most likely discover this at step 3 because the knowledge extractor will fail to obtain such values. There is, however, a small probability that V_P accepts and then tells V' what he wants to know. Although that probability is low, it is not zero. Hence, with very low probability, V could abuse P and learn something which he is not supposed to.

How is the simulator suppose to deal with this? Well, after discovering that V' is trying to cheat, S should simply try to finish the protocol (step 4) and hope to catch V' in the cheating. If V' is lucky, that is, if V' successfully cheats every round, S will be forced to output $\perp$ (that is S does not output a View) or S should go ahead and compute b' such that $(G_0, G_1) \approx (G_{b'}, G_{\overline{b'}})$.

In the latter case, the simulator will run in expected polynomial time since for all runs where V' gets caught cheating the simulator runs in polynomial time (as long as V' runs in polynomial time), but for the case in which V' can cheat without being caught by S, which happens with negligible probability (that is lower than 2^{-k}), then S will run in exponential time. If $k \in \omega(\log t(n))$, where n is the number of vertices of G_0 and G_1 and computing the isomorphism between G_0 and G'_b belongs to $\mathcal{O}(t(n))$, then on average, S would still run in expected polynomial time because $2^{-k}t(n)$ can be made less than 1 for an appropriate choice of k (see the analysis of Protocol 2.1).

In the former case, were S outputs $\perp$, the transcript where V' can cheat and P answers anyway will never be produced, hence $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ will not be distributed identically. Although the distance between the two distributions is negligible, this protocol would not qualify as Zero-Knowledge according to Definition 5. But it is very close to satisfying it. In fact we say that if the statistical distance between the two random variables is negligible, then the protocol is called statistical Zero-Knowledge. Let us define this new concept more formally.

3.2 Flavours of Zero-Knowledge

Definition 6 (Ensemble) *Let $\mathcal{I}$ be a countable set of indices. An Ensemble indexed by $\mathcal{I}$ is a sequence of random variables indexed by elements of $\mathcal{I}$.*

Namely, if $X = \{X_i\}_{i \in \mathcal{I}}$, where each X_i is a random variable, then X is an ensemble.

Definition 7 (Absolute (Perfect) Indistinguishability) *Let X and Y be two ensembles, then X and Y are perfectly indistinguishable if and only if*

$$\forall_h, \forall_n, \Pr[h(X_n) = 1] = \Pr[h(X_n) = 1], \tag{3.7}$$

where h is a predicate.

Definition 8 (Statistical Indistinguishability) *Let X and Y be two ensembles, then X and Y are statistically indistinguishable if and only if for all predicate h we have that*

$$|\Pr[h(X_n) = 1] - \Pr[h(Y_n) = 1]| \leqslant \mu(n) \tag{3.8}$$

where $\mu(n)$ is a negligible function.

Definition 9 (Computational Indistinguishability) *Let X and Y be two ensembles, then X and Y are computationally indistinguishable if and only if for all probabilistic polynomial time algorithm A we have that*

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leqslant \mu(n) \tag{3.9}$$

where $\mu(n)$ is a negligible function.

To each of these notions of indistinguishability corresponds a notion of Zero-knowledge: Perfect Zero-Knowledge, Statistical Zero-Knowledge and Computational Zero-Knowledge.

Going back to the Simulator for GNI, Simulator 3.8, we can conclude that Protocol 3.6 is a Statistical Zero-Knowledge protocol as the two random variables $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are identical but on points were S outputs $\perp$ which happens with negligible probability (see Problem 3.6).

If we consider n to be the number of vertices in G_0 or G_1, then if we fix k in protocol 3.6 to be in $\Theta(n)$, then the probability of failure of S is no more than $2^{-k} \leqslant 1/poly(n)$ for all $poly(n)$. And as was already argued, if S computes the isomorphism, then the protocol is a Perfect Zero-Knowledge protocol. The next Chapter will introduce Computational Zero-Knowledge protocols.

Problems

3.1 Prove formally that the simulator for Graph-Isomorphism is only expected-poly-time. Then fix the simulator so that it always runs in polynomial time. What can you conclude from your new simulator?

3.2 Prove formally that protocol 3.6 is a statistical-Zero-Knowledge protocol.

3.3
1. Prove that the proof of knowledge of Protocol 3.6, which we revisit at the end of this chapter, can be done in parallel. That is, if all pairs (G_0^i, G_1^i), all the challenges and all replies are each sent in one message so that the full proof of knowledge consists of three messages.
2. Is the Proof of knowledge still Zero-knowledge if done in parallel?
3. Prove that Protocol 3.6 is still zero-knowledge if the proof of knowledge is done in parallel.

3.4 Prove that for two ensemble X_n and Y_n that have negligible distance for all n, Theorem **??** implies that X_n and y_n are statistically indistinguishable (Definition 8).

3.5 quadratic residu.

1.

3.6 Prove that $D(VIEW[(P, V'(\aleph))(x)], S_{V'(\aleph)(x)})$ negligeable in the case were the Simulator outputs $\perp$. That is use convexity and the fact that the two random variables $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are the same with very high probability and very different with very low probability.

3.7 Let n be a product of two primes, and y be a quadratic non-residu $\bmod n$. Provide a protocol which is a Zero-Knowledge proof of Knowledge that a number x is a quadratic residu or a quadratic non-residu $\bmod n$. As your protocol is a proof of knowledge, it should not disclose whether x is or is not a residu, but only prove that the prover knows a witness to one or the other (it should also be simulatable).

Note that this is proving something *trivial* and is yet a very powerful tool. Can you find a situation where this could be used ?

3.8 We define **COCKS** as all the numbers n with two distinct prime factors p, q such that $\gcd(n, \phi(n)) = 1$, where $\phi(\cdot)$ is the euler totient function (so that n is invertible modulo $\phi(n)$ as required in Cocks' variation of the RSA cryptosystem using the public exponent $e = n$).

$$\mathbf{COCKS} = \{n | \gcd(n, \phi(n)) = 1, n = pq, p \neq q, p \text{ and } q \text{ are primes}\}.$$

Provide a Zero-Knowledge proof, with its simulator, to prove that a given integer n is a COCKS number, under the assumption that the verifier already knows an arbitrary $y \in QNR_n[+1]$[1].

Hint:

Let **WRSA** (Weak-RSA) be the following language:

$$\mathbf{WRSA} = \{n | n = p^\alpha q^\beta, p \neq q, p \text{ and } q \text{ are primes and } \alpha, \beta > 0\}.$$

A)** Construct a zero-knowledge proof for the language **WRSA**, under the assumption that the verifier already knows an arbitrary $y \in QNR_n[+1]$.

Sub-Hint: use Protocol 5.1. You may use the following theorem without proof:

Theorem 1 *If $n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$ (which is not a square) has exactly k distinct prime factors then exactly 2^{-k+1} of the elements in $\mathbb{Z}_n^*$ with Jacobi symbol $+1$ are quadratic residues $\bmod n$.*

Let **NphiN** be the following language:

$$\mathbf{NphiN} = \{n | \gcd(n, \phi(n)) = 1\}.$$

B) Construct a zero-knowledge proof for the language **NphiN**.

Sub-Hint: You may use the following theorem without proof:

Theorem 2 *Let n be a composite odd number. If n belongs to* **NphiN***, then every element $x \in \mathbb{Z}_n^*$ has an n^{th} root $\bmod n$, i.e. there exists an element y such that $x \equiv y^n \bmod n$. On the contrary, if n is not in* **NphiN***, then at most half the elements in $\mathbb{Z}_n^*$ have an n^{th} root $\bmod n$.*

C) Prove that we have

$$\mathbf{COCKS} = \mathbf{WRSA} \cap \mathbf{NphiN}.$$

D) Combine your two zero-knowledge proofs for languages **WRSA**, and **NphiN** to prove membership to **COCKS** in zero-knowledge, under the assumption that the verifier already knows an arbitrary $y \in QNR_n[+1]$.

3.9 Following on the previous question, define the following set of numbers

$$\mathbf{BLUM} = \{n | n = pq, p \equiv q \equiv 3 \bmod 4, p \neq q \text{ are primes}\}.$$

Prove that the language **BLUM** has a statistical zero-knowledge proof.

Hint:

[1]This technicality is necessary because we currently do not know an efficient algorithm to find such a y from n only.

Define the **Weak-BLUM** integers as

$$\mathbf{WBLUM} = \{n | -1 \in \mathrm{QNR_n}[+1]\}$$

Prove[2] that **BLUM** = **COCKS** ∩ **WBLUM**.

3.10 Revisit question 2.4 and make sure your protocol is Zero-Knowledge.

3.11 Revisit question 2.5 and make sure your protocol is Zero-Knowledge.

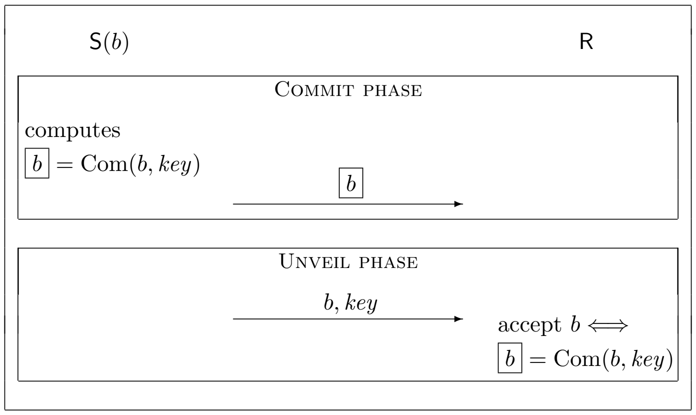
[2] In fact this statement is not 100% correct because **COCKS** excludes numbers $n = pq$ such that $p|q-1$ or $q|p-1$ while **BLUM** does not. However, in practice the difference is irrelevant.

Chapter 4

Bit-Commitment

A bit commitment scheme is a protocol between a sender S and a receiver R akin to locking a secret in a locked box and giving the box without the key to the receiver in a first phase (called **commit Phase**) and then in a second phase (called **unveil Phase**, or **unveiling**) to hand the key to the receiver.

In a more schematic way:



$\mathsf{S}(b)$
R
COMMIT PHASE
computes
$\boxed{b} = \mathrm{Com}(b, key)$
$\boxed{b}$
UNVEIL PHASE
b, key
accept $b \Longleftrightarrow$
$\boxed{b} = \mathrm{Com}(b, key)$

Protocol 4.1: General abstract bit-commitment

Here $\boxed{b}$ is often called a blob. It is a string of bits which is essentially an encryption of b (albeit with one more property that we shall see below). Note that in general there might be a more or less long period of time between the commit phase and the unveil phase. The *key* variable is simply some side data needed to do the commitment, usually some side randomness. There are two main properties that a bit commitment should have :

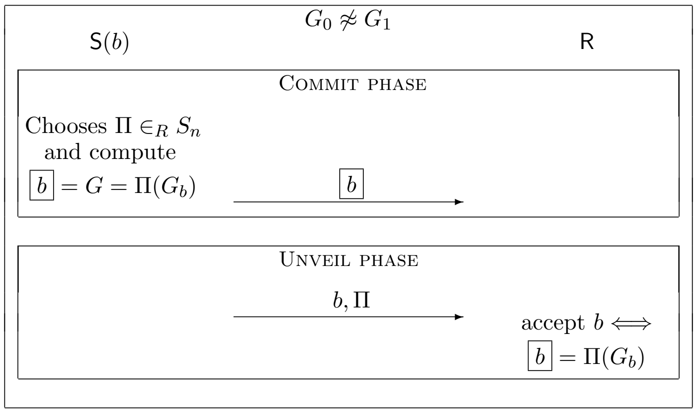
First, a **concealing property**: the receiver is unable to tell whether the Sender committed to b or $\bar{b}$ until the unveil phase. This is essentially akin to a Semantically secure encryption of b.

Second, a **binding property**: that is the sender cannot change the bit b to $\bar{b}$ between the reception of $\boxed{b}$ by R and the unveil phase. This is a property that encryption schemes do not usually have. More mathematically:

$$\begin{array}{ll} \text{Binding:} & \{\mathrm{Com}(b, key)\}_{key} \cap \{\mathrm{Com}(\bar{b}, key)\}_{key} = \emptyset \\ \text{Concealing:} & P_{\mathrm{Com}(b,key)} =_0 P_{\mathrm{Com}(\bar{b},key)}, \end{array} \tag{4.1}$$

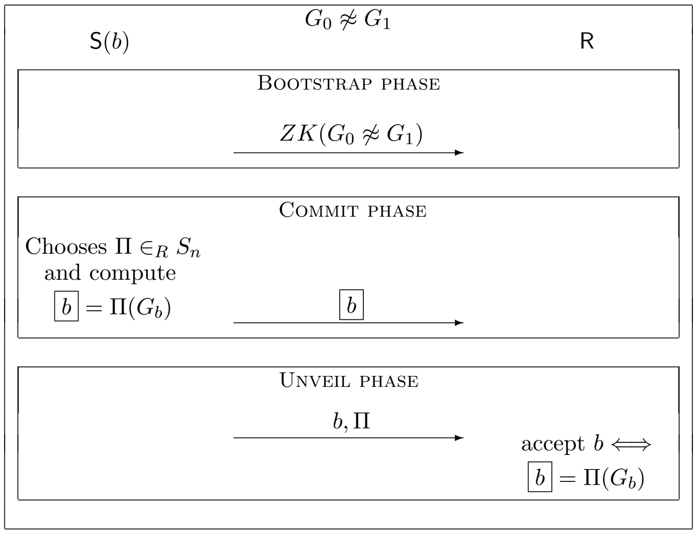
where $P_{\text{Com}(b,key)}$ is the distribution of the possible bit commitment for b using the side information key and $\{\text{Com}(b, key)\}_{key}$ is simply the set of all possible bit-strings that can be commitment to b indexed by the possible key's.

As stated in (4.1), the binding and conceilling properties are mutually exclusive. How can two random variables be equal (hence undistinguishable) if they are defined on orthogonal sets? We shall water down the definition in order to reach it. Let us do that by looking at a few examples.



Protocol 4.2: Bit commitment based on GNI

It is impossible for the sender to change $\boxed{b}$ to $\boxed{\bar{b}}$ as this would tantamount to finding a permutation ρ that maps $\Pi(G_b)$ to $G_{\bar{b}}$, and we know that such a permutation does not exist. Or do we? How can the receiver be sure of that? For that matter, the Sender would need to prove to him that indeed the graphs are not isomorphic. We should thus revisit our protocol like this:



Protocol 4.3: BC based on GNI with bootstrap

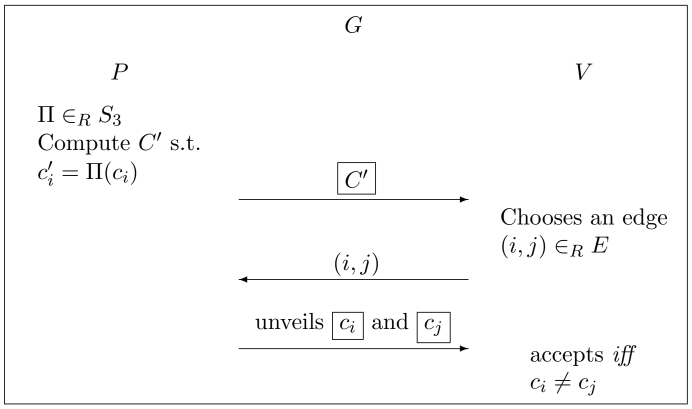
Now the receiver is convinced that the two graphs are not isomorphic, thus the receiver is convinced that the binding property of the bit-commitment will hold. The concealing property of this bit-commitment cannot be stronger than computational, as a more powerful receiver might go ahead and compute an isomorphism between $\boxed{b}$ and G_c for some c. But we assume R to be probabilistic-polynomial time, hence if the pair of graph (G_0, G_1) is well chosen, then the protocol is computationally conceiling[1]. Note that to prove this, one needs to reduce the problem GNI to an adversary that could break the concealing property.

4.1 Computationally Concealing (CC)BC and Computational ZK

Before going forward with commitment schemes, let us see how to use them in a cryptographic setting. Here is a computational zero-knowledge protocol for an NP-Complete problem: **3-Coloring**.

For this protocol, we extend the notion of a graph so that each vertex also holds a color. Let us have 3 colors, red, green, blue: $Col = \{\text{red}, \text{green}, \text{blue}\} = \{00, 01, 10\}$, and 11 be an invalid color.

So the colored graph G is $G = \langle \{c_v\}_{v \in V}, E\rangle$ and for all u, v if $(u, v) \in E$ then we have that $c_u \neq c_v$. Hence a graph is now a vector of colors and an adjacency matrix. Let $C = (c_1, c_2, \ldots, c_n)$ be the vector of colors for the graph G. Here is the protocol:



Protocol 4.4: A Zero-Knowledge protocol for 3-Col

The first thing to mention is the notion of a commitment to such a large object as C'. The vector C' is composed of $2n$ bits if there are n vertices in the graph. Hence $\boxed{C'}$ is really constituted of $2n$ standard bit-commitments. Two bit-commitments per color. If $c'_i = 01$, then $\boxed{c'_i} = \boxed{0}\boxed{1}$. From this, we conclude that unveiling $\boxed{c_i}$ and $\boxed{c_j}$ is as easy as unveiling the four distinct bit-commitments.

So we only need to verify the completeness and soundness properties and then provide a simulator.

- if $G \in$ 3-Col: $\Pr[(P, V)(G) = 1] = 1$
- if $G \notin$ 3-Col: $\Pr[(P', V)(G) = 1] \leqslant \frac{|E|-1}{|E|} = 1 - \frac{1}{|E|}$.

If the graph is not 3-colorable, then it must be that for at least one edge, the vertices of that edge are labeled with the same color. If V is honest, then V will choose the edge (v_i, v_j) at rancom, hence V has probability at least $1/|E|$ of choosing an edge for which the coloring is not valid. If there are more than one edge for which the coloring is invalid, then the probability can only be higher. So we only need to amplify this probability in order to get a valid interactive proof.

[1]For as long as there exists a class of Graphs for which GI is hard.

We know that $|E| < n^2$, where $n = |V|$. If we repeat k times the protocol, where $k = |E| \in \mathcal{O}(n^2)$ we get that

$$\Pr[(P', V)(G) = 1] \leqslant \left(1 - \frac{1}{|E|}\right)^k = \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e} \tag{4.2}$$

were we have used Formula 4. Hence we get an interactive proof. If we repeat a factor of n times more this composite protocol, then we get a negligible probability of error, that is, if $k \geq n^3$ we have

$$\Pr[error] = \Pr[(P', V)(G) = 1 | G \notin \text{3-Col}] \leqslant \frac{1}{e^n}. \tag{4.3}$$

We can now present a simulator and thus prove that the protocol is zero-knowledge.

```
1. Initialize $V$:  copy fresh random bits to the verifier's random tape and
   fill-up the Auxiliary-Input tape.
2. Pick a random $\Pi \in_R S_3$ and a random edge $(i', j') \in_R E$.  Compute $C''$ s.t.
   $c''_{i'} = \Pi(\texttt{red}), c''_{j'} = \Pi(\texttt{green}), c''_{k'} = \Pi(\texttt{blue})$ for all $k' \neq i', k' \neq j'$.
3. Commit to that vector by giving $\boxed{C''}$ to $V$.
4. Wait for V to answer with an edge $(i, j)$
5. (a) if $i' = i$ and $j' = j$, then write transcript,
   (b) else rewind to Step 2.
```

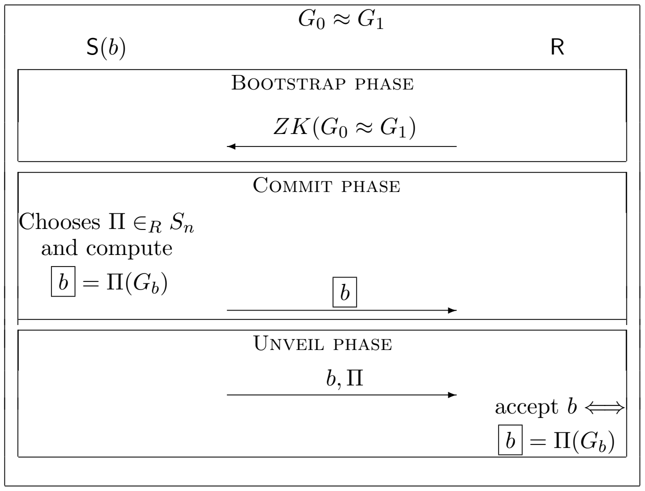
Simulator 4.5: Simulator for Graph 3-Coloring

By the properties of the bit commitment, a vector of colors generated by the simulator is computationally indistinguishable from a vector generated by a genuine prover. Hence, V cannot see a difference between the bogus vector to which S is committed and a real one as the Random Variables $\boxed{C'}$ and $\boxed{C''}$ are computationally indistinguishable. Hence, the forged transcript will be computationally indistinguishable from a real one. The output of V being based on computationally indistinguishable random variables, it must also be computationally indistinguishable between the run with the real prover and the run with the simulator. Hence, we have a computational zero-knowledge proof system for 3-Col that runs in expected polynomial time, as the probability that i', j' is equal to i, j is $1/|E|$, thus on average, S will run the simulation $|E|$ times before being able to write a triple $\left(\boxed{C''}, (i, j), \text{unveils}(c''_i, c''_j)\right)$ to the transcript ; $|E| < n^2$, where n is the number of vertices in the graph.

Note, that to any one with the capacity to break the concealing property of the bit-commitment, it will be obvious that the simulator generated transcript is not a real one. The protocol would not be zero-knowledge anymore, as that better-than-PPT-verifier could just read the coloring in the prover's commitment to C'.

4.2 Computationally Binding BC

Let us now look at a different flavor of bit-commitments, that is a commitment which is perfectly conceiling, but computationally binding.



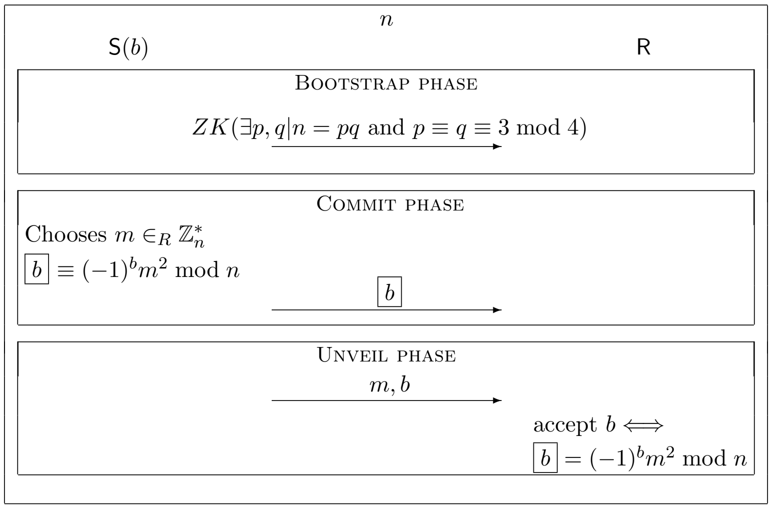
Protocol 4.6: Bit commitment based on GI

This protocol looks like a mirror image of Protocol 4.3. The main difference is that the two public graphs G_0 and G_1 are now isomorphic and that the commitment completely hides the bit b, whilst the sender is only computationally bounded to his bit b. Protocols of this type and their applications will be the subject of next chapter.

4.3 CCBC from standard Cryptographic Assumptions

Here is a more practical example (similar to Protocol 4.3) based on the problem of deciding quadratic residuosity. It is more practical in the sense that S can peform the entire protocol in polynomial-time and for practical sizes of the parameter n, the resulting Bit Commitment is currently hard to break whereas in Protocol 4.3 it is not clear that we can find graphs for which GI is hard to decide...

For p and q congruent to $3 \mod 4$, we know that -1 is a quadratic non-residue with Jacobi symbol $+1$ modulo $n = pq$ and furthermore, computing square roots is easy for anyone who knows p and q see Problem 3.5.


n
S(b)
R
Bootstrap phase
$ZK(\exists p, q | n = pq \text{ and } p \equiv q \equiv 3 \bmod 4)$
Commit phase
Chooses $m \in_R \mathbb{Z}_n^*$
$\boxed{b} \equiv (-1)^b m^2 \bmod n$
$\boxed{b}$
Unveil phase
m, b
accept $b \iff \boxed{b} = (-1)^b m^2 \bmod n$

Protocol 4.7: Bit commitment based on Quadratic reduosity

Because -1 is a non-residue with Jacobi symbol $+1$, we know that $(-1^1)m^2$ is also a non-residue and that $(-1^0)m^2$ is a residue, both with Jacobi symbol $+1$. The two sets being disjoint, the protocol is obviously statistically[2] binding . As for the concealing property, distinguising between random non-residues and residues is believed to be hard. Problem 4.3 asks you to prove formally that the protocol is concealing.

4.4 Hard-core predicate

Definition 10 (Hard-Core predicate) *A polynomial time computable predicate $g : \{0,1\}^* \longrightarrow \{0,1\}$ is called a hard-core predicate for the function f if for every polynomial-time probabilistic algorithm A' and every sufficiently large n we have*

$$\left|\Pr_m\left[A'(f(m)) = g(m)\right] - \frac{1}{2}\right| \leqslant \mu(n) \tag{4.4}$$

where $\mu(\cdot)$ is a negligible function and the m are chosen with uniform probability.

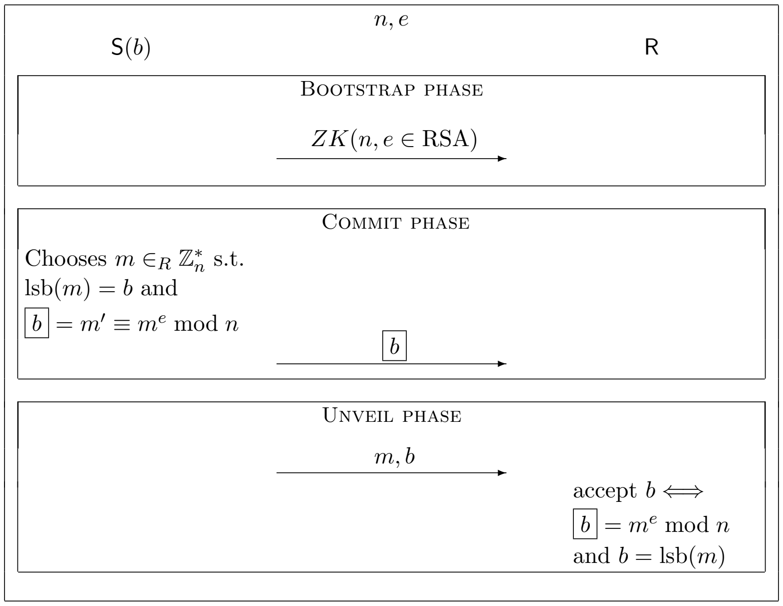
In layman's terms, this says that it is hard on average for A' to guess $g(x)$ given $f(x)$ (that is without seeing x itself), if x is chosen uniformly in its domain. If the function f is injective, then the existence of a hard-core predicate of f implies that f is a one-way function.

Note that this definition also implies that the bias of the predicate g cannot be to large, or

$$|\Pr[g(U_n) = 0] - \Pr[g(U_n) = 1| < \mu'(n) \tag{4.5}$$

for some other negligible function $\mu'(\cdot)$.

Using hard-core predicate we can build perfectly binding and computationally concealing bit-commitment. Here is a first example using RSA.



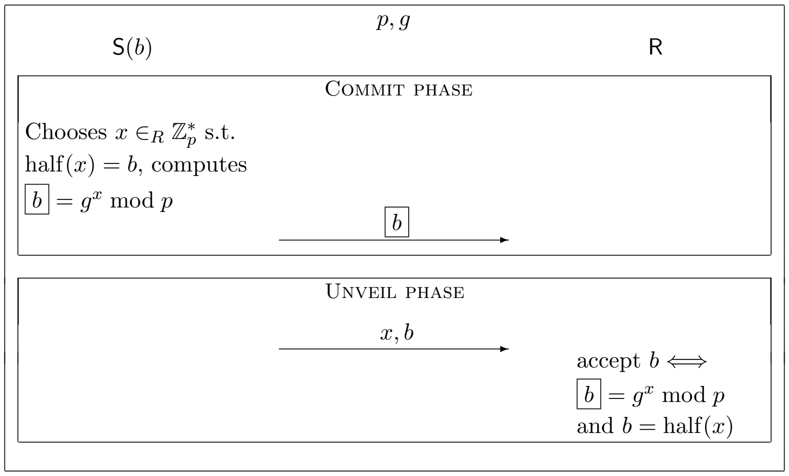
n, e
S(b)
R
Bootstrap phase
$ZK(n, e \in \text{RSA})$
Commit phase
Chooses $m \in_R \mathbb{Z}_n^*$ s.t. $\text{lsb}(m) = b$ and $\boxed{b} = m' \equiv m^e \bmod n$
$\boxed{b}$
Unveil phase
m, b
accept $b \Longleftrightarrow \boxed{b} = m^e \bmod n$ and $b = \text{lsb}(m)$

Protocol 4.8: Bit commitment based on RSA

[2] it is not perfect because with exponentially small probablity, an n of the wrong form may be accepted by the verifier and -1 can be a quadratic residue. Consequently the commitment may not be binding at all.

Here $\text{lsb}(m)$ is the least significant bit of the string m. Note that RSA is a one-way permutation for all $m \in \mathbb{Z}_n^*$ and that for RSA the predicate $\text{lsb}(m)$ is considered intractable given m^e. Note also that for this to work, p, q (where $n = pq$) and d , the decryption exponent, are to be kept secret from the receiver. Only n and e are public, see problem **??** for the Bootstrap phase.

Here is a more reasonable example using the discrete logarithm.



p, g
$\mathsf{S}(b)$
R
COMMIT PHASE
Chooses $x \in_R \mathbb{Z}_p^*$ s.t.
$\text{half}(x) = b$, computes
$\boxed{b} = g^x \bmod p$
$\boxed{b}$
UNVEIL PHASE
x, b
accept $b \iff$
$\boxed{b} = g^x \bmod p$
and $b = \text{half}(x)$

Protocol 4.9: Bit commitment based on Discrete Logarithm

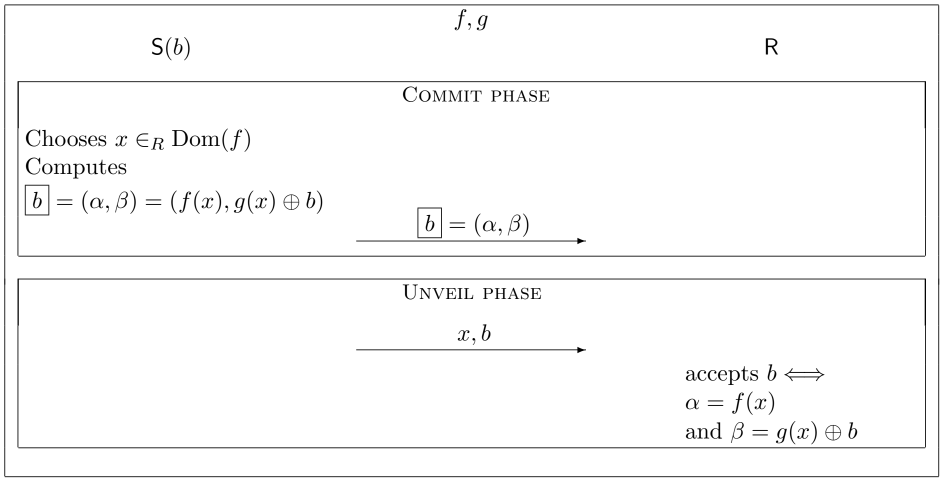
For Protocol 4.9, g is a generator for the multiplicative group $\mathbb{Z}_p^*$ and the hard-core predicate half is defined as follows:

$$\text{half}(x) \quad = \quad \begin{cases} 0 & \text{if} \quad 1 \leqslant x \leqslant \frac{p-1}{2} \\ 1 & \text{if} \quad \frac{p-1}{2} < x \leqslant p-1, \end{cases}$$

which is intractable if the discrete log is a one way function.

The constructions of bit-commitment based on hard-core predicate we have seen so far use the algebraic notions of the intractable problem used and the properties of the hard-core predicate itself. But there is a way to construct bit-commitment schemes that will work for any hard-core predicate and which does not rely on any algebraic structure for its security.

Let $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ be a one-way permutation and g a hard-core predicate of the permutation f and let $\text{Dom}(f)$ be the domain of the function f. Protocol 4.10 is a general bit-commitment scheme that only uses the fact that f is a one-way permutation.


f, g
$\mathsf{S}(b)$
R
COMMIT PHASE
Chooses $x \in_R \mathrm{Dom}(f)$
Computes
$\boxed{b} = (\alpha, \beta) = (f(x), g(x) \oplus b)$
$\boxed{b} = (\alpha, \beta)$
UNVEIL PHASE
x, b
accepts $b \Longleftrightarrow$
$\alpha = f(x)$
and $\beta = g(x) \oplus b$

Protocol 4.10: Bit commitment based on any hard-core predicate

Now is a good time to remind the reader of the definition of a one way function (or permutation).

Definition 11 (One-Way function) *A function $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ is said to be one-way if the two following conditions hold.*

1. *There exists an probabilistic-polynomial time algorithm A such that $f(x) = A(x)$ for all x in the Domain of the function f,*

2. *For all probabilistic-polynomial time algorithm A' we have*

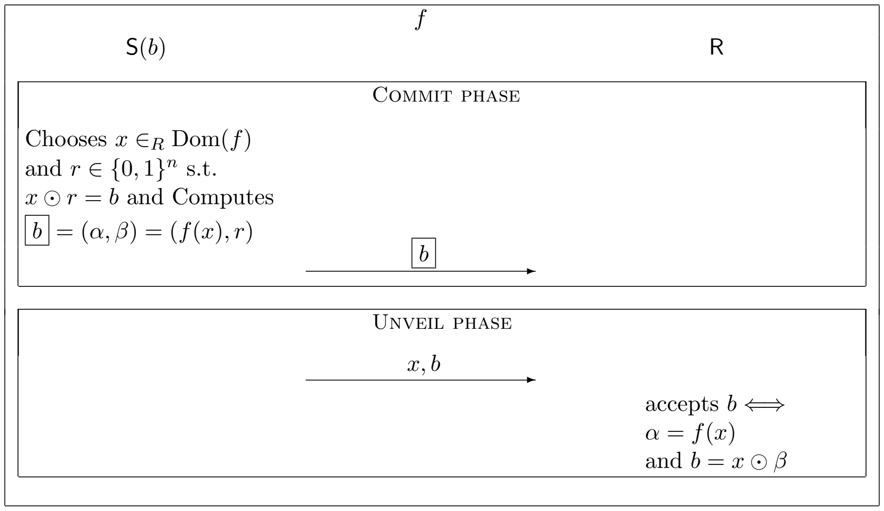
$$\Pr_x[A'(f(x), 1^n) \in f^{-1}(f(x))] \leqslant \mu(n), \tag{4.6}$$

where x is chosen at random uniformly in the Domain of f and μ is a negligible function.

What if a one-way function does not have any obvious hard-core predicate. Does there exists a general one that we could use for any given function? Well yes, as the next Theorem states. First let $x = x_1 x_2 \dots x_n$ and $r = r_1 r_2 \dots r_n$ be two n-bit strings. Then we write the inner product between the two strings as $x \odot r \triangleq \bigoplus_i x_i \wedge r_i = (\sum_i x_i \cdot r_i) \bmod 2$.

Theorem 3 (Goldreich-Levin) *Let f be a one-way function with domain over n-bit strings and let the function f' be defined by $f'(x, r) \triangleq (f(x), r)$, where $|x| = |r|$. Define the predicate $g(x, r) \triangleq x \odot r$, then g is a hard-core predicate of the function f'.*

Clearly, if f is a one-way function(permutation) then f' is a one-way function(permutation) too. Using this general hard-core predicate and a general one-way permutation, we can do bit-commitment.


f

$\mathsf{S}(b)$ R

COMMIT PHASE

Chooses $x \in_R \mathrm{Dom}(f)$
and $r \in \{0,1\}^n$ s.t.
$x \odot r = b$ and Computes
$\boxed{b} = (\alpha, \beta) = (f(x), r)$

$\xrightarrow{\boxed{b}}$

UNVEIL PHASE

$\xrightarrow{x, b}$

accepts $b \Longleftrightarrow$
$\alpha = f(x)$
and $b = x \odot \beta$

Protocol 4.11: Bit commitment based on Goldreich-Levin predicate

4.5 Pseudorandom Generators

Definition 12 (Pseudorandom Generators) *An algorithm $G : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$, deterministic polynomial time, where $l(n) > n$, is a pseudorandom generator if for every probabilistic polynomial time algorithm A' we have*

$$\left|\mathrm{Pr}_x[A'(G(x)) = 1] - \mathrm{Pr}_y[A'(y) = 1]\right| < \mu(n), \tag{4.7}$$

where, $x \in_R \{0,1\}^n$, $y \in_R \{0,1\}^{l(n)}$ and $\mu(n)$ is a negligible function.

The function $l(\cdot)$ is called the expansion factor of G.

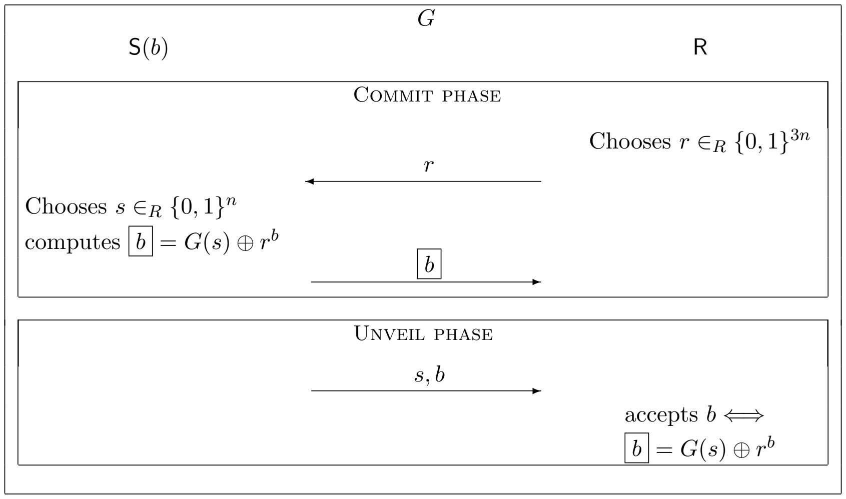
Theorem 4 (Pseudorandom Generator amplification) *Let $G : \{0,1\}^n \longrightarrow \{0,1\}^{n+1}$ be a pseudo random generator and $l(\cdot)$, where $l(n) > n$, be a function, then a pseudorandom Generator $G' : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ can be constructed from G.*

Generator G' will be constructed by by calling multiple times the generator G and then combining all these calls output into one output for G'. Let $G(s_i) = s_{i+1}||\sigma_{i+1}$ where σ_{i+1} is a one-bit string and s_i and s_{i+1} are n-bit strings. And concatenation of string is denoted by the the symbol $||$. The output of $G'(s_0)$ is then $\sigma_1||\sigma_2||\ldots||\sigma_{l(n)}$.

We can build a $n+1$ pseudogenerator using the discrete logarithm. Let g be a generator of the group, then $G(s) = (g^s \bmod p)||\mathrm{half}(s)$. Hence, using the construction above, we can construct a pseudorandom generator G' based on $G(s)$, $G'(s_0) = \mathrm{half}(s_0)||\mathrm{half}(s_1)||\ldots||\mathrm{half}(s_{l(n)-1})$, which is just a succession of hard-core predicate. In fact, this construction using hard-core predicate can always be done for any one way permutation. The following is a very important result about pseudo-random generator.

Theorem 5 *Pseudorandom generators exist if and only if one-way functions exist.*

Pseudorandom generators have many uses, in particular, one can build bit-commitment from them. Here is the construction of a general bit-commitment from any pseudorandom generator whose output length is three times as long as it input length: $|G(s)| = 3|s|$.


G
$\mathsf{S}(b)$
R
COMMIT PHASE
Chooses $r \in_R \{0,1\}^{3n}$
r
Chooses $s \in_R \{0,1\}^n$
computes $\boxed{b} = G(s) \oplus r^b$
$\boxed{b}$
UNVEIL PHASE
s, b
accepts $b \iff$
$\boxed{b} = G(s) \oplus r^b$

Protocol 4.12: Bit commitment based on Pseudorandom generator

Concealment follows from the property of pseudorandom generator, that is, $G(s)$ is computationally undistinguishable from the uniform distribution over $3n$ bits. The uniform distribution xored with a fixed string r is itself the uniform distribution which is undistinguishable from $G(s) \oplus r$. Hence, $G(s)$ and $G(s) \oplus r$ are indistinguishable, as indistinguishability is a transitive property, see Problem 5.10.

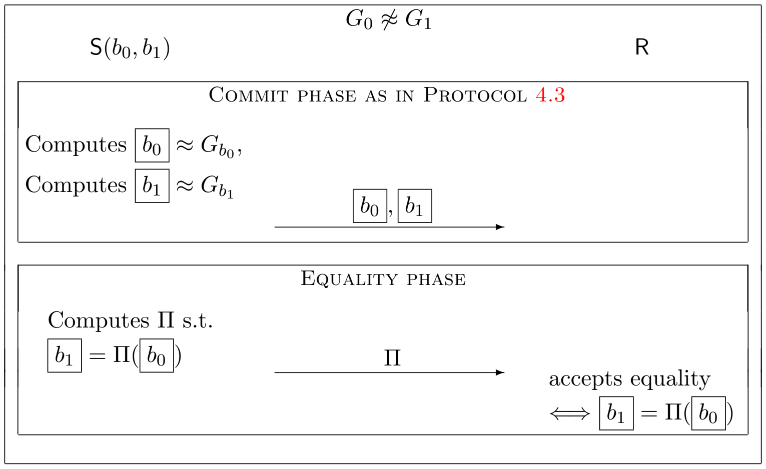
Binding comes from the fact that the image of $G(s)$ is very sparce, or $\Pr[\exists s' \text{ s.t. } G(s) = G(s') \oplus r]$ is low. It is possible (not necessarily easily) to cheat if and only if $\exists s'$ s.t. $G(s) \oplus G(s') = r$. The number of pairs (s, s') is easy to compute, it simply $\binom{2^n}{2} < 2^{2n}$ whilst the number of dirrent r is of course 2^{3n}. Hence

$$\Pr[R \text{ chooses } r \text{ s.t. } \mathsf{S} \text{ cannot cheat}] > 1 - 2^{2n-3n} = 1 - 2^{-n}.$$

Hence binding is statistical and concealment is computational.

4.6 Bit-commitment equality

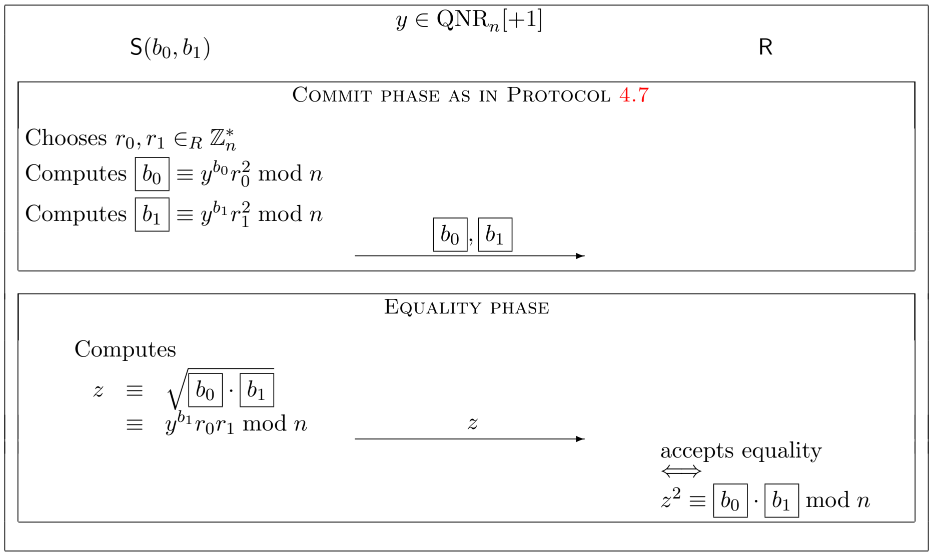
Until now, commitments have being considered as blobs that behave as locked-boxes. To do useful computation, we need more properties. One important property will be to compare commitments and prove that two commitments in fact hide the same value without revealing that value. Here is a simple example based on Graph Non-Isomorphism.


$G_0 \not\approx G_1$
$\mathsf{S}(b_0, b_1)$
R
COMMIT PHASE AS IN PROTOCOL 4.3
Computes $\boxed{b_0} \approx G_{b_0}$,
Computes $\boxed{b_1} \approx G_{b_1}$
$\boxed{b_0}, \boxed{b_1}$
EQUALITY PHASE
Computes Π s.t.
$\boxed{b_1} = \Pi(\boxed{b_0})$
Π
accepts equality
$\iff \boxed{b_1} = \Pi(\boxed{b_0})$

Protocol 4.13: Bit commitment Equality using GNI

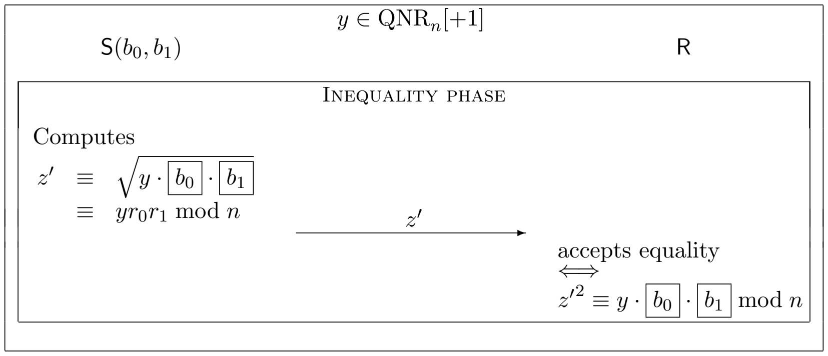
If, in the bootstrap phase, the two graphs G_0 and G_1 are shown to be non isomophic, then if $\boxed{b_0}$ and $\boxed{b_1}$ are commitments to different bits, then no permutation Π mapping one to the other should exist. Hence this protocol can succeed if and only if the two commitments are in fact to the same bit: $b_0 = b_1$ and $\boxed{b_0} \approx \boxed{b_1}$. Of course, if the receiver cannot distinguish between commitments to a zero and commitments to a 1, then after seeing a permutation between the two commitments, the receiver cannot tell whether they are two commitments to a zero or to a one (Otherwise, one could build a concealing breaking adversary from such an adversary), see Problem 3.5

Here is a more practical example using quadratic non-residues.


$y \in \mathrm{QNR}_n[+1]$
$\mathsf{S}(b_0, b_1)$
R
COMMIT PHASE AS IN PROTOCOL 4.7
Chooses $r_0, r_1 \in_R \mathbb{Z}_n^*$
Computes $\boxed{b_0} \equiv y^{b_0} r_0^2 \bmod n$
Computes $\boxed{b_1} \equiv y^{b_1} r_1^2 \bmod n$
$\boxed{b_0}, \boxed{b_1}$
EQUALITY PHASE
Computes
$z \equiv \sqrt{\boxed{b_0} \cdot \boxed{b_1}}$
$\equiv y^{b_1} r_0 r_1 \bmod n$
z
accepts equality
$\iff$
$z^2 \equiv \boxed{b_0} \cdot \boxed{b_1} \bmod n$

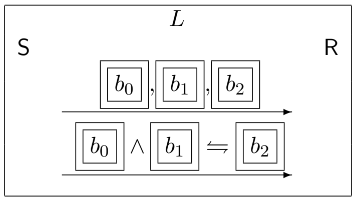
Protocol 4.14: Bit commitment Equality using QNR

If the two commitments $\boxed{b_0}$ and $\boxed{b_1}$ are not to the same bit, then the equality phase cannot be won by S as a quadratic residue multiplied by a quadratic non-residue always gives a quadratic non-residue. Hence this equality proof is correct. But this last observation gives us the means to also easily prove inequality between $\boxed{b_0}$ and $\boxed{b_1}$. Let the two commit phases in Protocol 4.15 be as in Protocol 4.14.


$y \in \mathrm{QNR}_n[+1]$
$\mathsf{S}(b_0, b_1)$
R
INEQUALITY PHASE
Computes
$z' \equiv \sqrt{y \cdot \boxed{b_0} \cdot \boxed{b_1}}$
$\equiv y r_0 r_1 \bmod n$
z'
accepts equality
$\Longleftrightarrow$
$z'^2 \equiv y \cdot \boxed{b_0} \cdot \boxed{b_1} \bmod n$

Protocol 4.15: Bit commitment Inequality using QNR

From here on, we shall denote a bit commitment scheme equiped with equality as $\boxed{\boxed{b}}$ and the action of proving equality between two such commitements $\boxed{\boxed{b_0}}$ and $\boxed{\boxed{b_1}}$ by $\boxed{\boxed{b_0}} \rightleftharpoons \boxed{\boxed{b_1}}$, for whatever commitment scheme used. Now that we know how to prove equality between bit commitments, we shall show how to compute boolean circuits on committed inputs. For example, here is an abstract version of what we shall show for the AND gate.

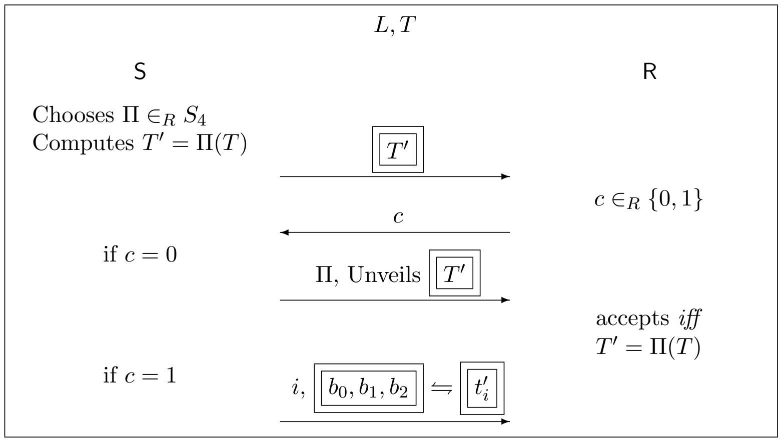

L
S
R
$\boxed{\boxed{b_0}}, \boxed{\boxed{b_1}}, \boxed{\boxed{b_2}}$
$\boxed{\boxed{b_0}} \wedge \boxed{\boxed{b_1}} \rightleftharpoons \boxed{\boxed{b_2}}$

Protocol 4.16: Abstract protocol for boolean circuit

We shall prove this using a truth table. For example, for the AND gate, we have the following table $T = (t_0, t_1, t_2, t_3)^T$, where $t_i = (t_{i,0}, t_{i,1}, t_{i,2})$:

$$\begin{aligned} t_0 &: \quad 0 \wedge 0 = 0 \\ t_1 &: \quad 0 \wedge 1 = 0 \\ t_2 &: \quad 1 \wedge 0 = 0 \\ t_3 &: \quad 1 \wedge 1 = 1 \end{aligned}$$

To prove that the three bit-commitments obey some boolean gate relation, the sender will choose a random permutation over S_4 and permute the rows of T, that is $\Pi(T) = \Pi(t_0, t_1, t_2, t_3) = (t'_0, t'_1, t'_2, t'_3) = T'$, and send a committed permuted table to the receiver. The receiver will chose a bit c at random and send it to the receiver who will then, depending on the bit c, either unveil the permuted table to show that he was committed to a valid table or prove that the three bit-commitments are equal bit by bit to one of the rows of the table.


L, T
S
R
Chooses $\Pi \in_R S_4$
Computes $T' = \Pi(T)$
$\boxed{\boxed{T'}}$
$c \in_R \{0,1\}$
c
if $c = 0$
Π, Unveils $\boxed{\boxed{T'}}$
accepts *iff* $T' = \Pi(T)$
if $c = 1$
$i, \boxed{\boxed{b_0, b_1, b_2}} \leftrightharpoons \boxed{\boxed{t'_i}}$

Protocol 4.17: Zero-knowledge computation of a boolean gate

Here, in Protocol 4.17, $\boxed{\boxed{b_0, b_1, b_2}}$ is the triplet of the three individual commitment for b_0, b_1 and b_2; $\boxed{\boxed{t'_i}}$ is also a triplet of commitments and $\boxed{\boxed{b_0, b_1, b_2}} \leftrightharpoons \boxed{\boxed{t'_i}}$ means running three times the equality part of Protocol 4.14 for example with the pairs $\left(\boxed{\boxed{b_0}}, \boxed{\boxed{t'_{i,0}}}\right)$, $\left(\boxed{\boxed{b_1}}, \boxed{\boxed{t'_{i,1}}}\right)$ and $\left(\boxed{\boxed{b_2}}, \boxed{\boxed{t'_{i,2}}}\right)$, which would reveal nothing but that the three commitments to b_0, b_1 and b_2 do obey one of the row of the table T'.

If the sender S is commited to three bits which in fact obey a specific boolean gate relation represented by T, then the sender will always pass this protocol. What if S is cheating. Then, it means, that for any valid T', there is no row in T' that can be proven equal to the three bits b_0, b_1 and b_2. Hence if the sender tries to cheat, then he has two choices: either he commits to a valid T' and he gets caught with probability one half if the receiver ask to see the proof of equality (although he will pass the test if $c = 0$ with probability one half); or he commits to a table T' which is not a permutation of T, which in this case will let him pass successfully the test if $c = 1$, which also happens with probability one half. In any case, if S is cheating, he will get caught with probability one-half. So Protocol 4.17 is definitely a valid interactive proof to show that a triplet obeys a certain boolean relationship defined by T.

We only need to provide a simulator in order to prove that this protocol is indeed Zero-knowledge. The simulator for Protocol 4.17 ressembles Simulator 3.5: that is the simulator first choses to which queries he wishes to answer and prepares for that eventuality. If the Receiver asks the question for which the simulator was prepared, then the Simulator happily writes down the transcript otherwise the simulator resets R and prepares for a new question. The idea being that from the bit commitments $\boxed{\boxed{T'_j}}$ and $\boxed{\boxed{b_0 b_1 b_2}}$ the receiver does not know that these do not constitute a valid IP system; the real bits being hidden by the commitment protocol, the Receiver's choice cannot be biased toward detecting that the simulator is cheating. Hence, the probability that the simulator can cheat and can write a transcript is about one half. Note, that in a larger protocol, step one of Simulator 4.6 would be different; it would probably not exists as the receiver would have already been initialized and the values $\boxed{\boxed{b_0}}$, $\boxed{\boxed{b_1}}$ and $\boxed{\boxed{b_2}}$, as table T, would already have been fixed. This protocol can implement any boolean gate $(\neg, \wedge, \vee, \oplus, \ldots)$, as table T only has to be correctly defined and the protocol is easily adapted to unary gates.

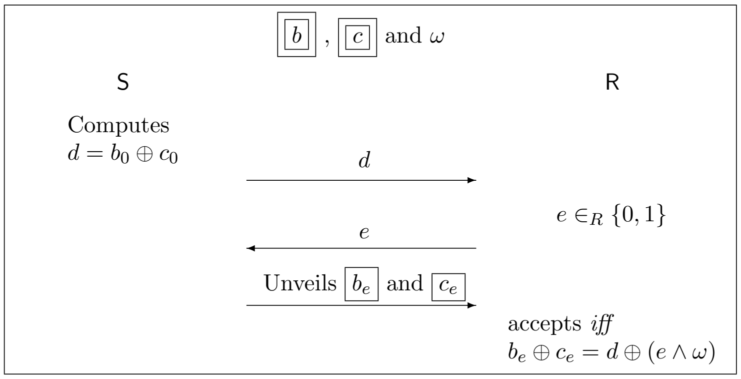
```
1. Copy fresh random bits to the receiver's random tape, fill up the
   Auxiliary-Input tape and fix table $T$ and commitments $\boxed{\boxed{b_0}}$, $\boxed{\boxed{b_1}}$ and $\boxed{\boxed{b_2}}$.
2. Pick a bit $c' \in_R \{0,1\}$ and permutation $\Pi \in_R S_4$.
3. (a) If $c' = 0$ compute $T' = \Pi(T)$
   (b) If $c' = 1$ compute $T' = \Pi(T'')$, where $T''$ is the table $T$ with the row $t_0$
       changed to $(b_0, b_1, b_2)$
4. Run the receiver R on input $\boxed{\boxed{T'}}$ and wait for his reply $c$.
5. (a) If $c = c'$, then finish simulation
   (b) if $c \neq c'$, then rewind R as if nothing had happened and go back to step 2
```

Simulator 4.18: Simulator for table T

4.6.1 Rudich's Trick

What if the bit-commitment protocol we wish to use does not naturally provide an easy way to prove equality between two commitments? Think of the bit commitment based on the hard-core predicate "half" and the discrete logarithm problem. How can one easily compare two bit-commitments based on "half"? In general, there are no theoretical arguments arguing for the existence of such a procedure for every bit-commitment protocol. Yet, we can do something thanks to Rudich's trick.

The trick is to represent a commitment to a bit b by using two commitments to random bits b_0 and b_1 such that $b = b_0 \oplus b_1$. Hence a rudich's commitment to b, which we shall denote $\boxed{\boxed{b}}$, is simply the pair $\boxed{\boxed{b}} = \left(\boxed{b_0}, \boxed{b_1}\right)$ and the commitment flavor used is whatever commitment scheme that is available. The next protocol allows the sender to prove to the receiver that two commitments $\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$ are equal if ω is zero and unequal if ω is one.


$\boxed{\boxed{b}}$, $\boxed{\boxed{c}}$ and ω
S
R
Computes
$d = b_0 \oplus c_0$
d
$e \in_R \{0,1\}$
e
Unveils $\boxed{b_e}$ and $\boxed{c_e}$
accepts *iff*
$b_e \oplus c_e = d \oplus (e \wedge \omega)$

Protocol 4.19: Destructive equality for Rudich's commitment

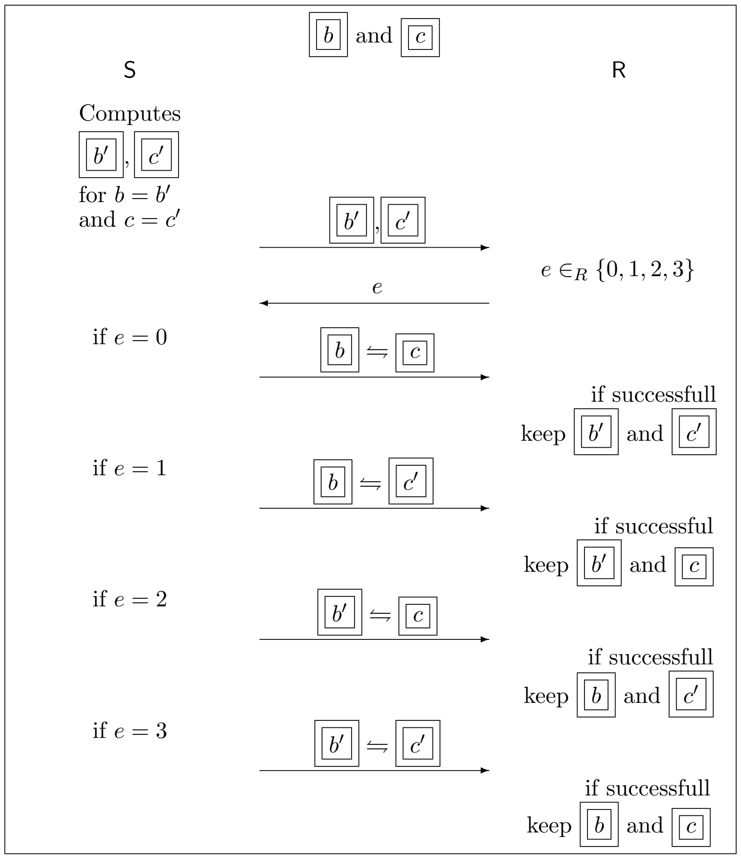
The following table proves that protocol 4.19 is complete and has soundness one half.

$b = c$						$b \neq c$					
b_0	b_1	c_0	c_1	$b_0 \oplus c_0$	$b_1 \oplus c_1$	b_0	b_1	c_0	c_1	$b_0 \oplus c_0$	$b_1 \oplus c_1$
0	0	0	0	0	0	0	0	0	1	0	1
0	0	1	1	1	1	0	0	1	0	1	0
0	1	0	1	0	0	0	1	0	0	0	1
0	1	1	0	1	1	0	1	1	1	1	0

Table 4.1: Visual proof of correctness

Of course, Table 4.1 represents only half of the possibilities that b and c can take, but the second half is obtained by taking the complement of every bit b_i and c_i in the table. From this table we conclude that if S is honest and that $\boxed{\boxed{b}}$ is a commitment to the same value as $\boxed{\boxed{c}}$, then R will always accept at the end. Whilst if $\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$ are not commitments to the same value, then whatever value d is sent in the first message, with probability one half, S will be asked to open a pair of value that do not xor to it.

But this protocol has a slight problem, it has destroyed the commitments. Hence we know that $\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$ were commitments to the same value, but S cannot use them any more (See Problem 4.10). We need to be a little more creative. Consider the next protocol, Protocol 4.21 in which two new commitments are introduced and one pair shall survive at the end.


$\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$
S
R
Computes
$\boxed{\boxed{b'}}, \boxed{\boxed{c'}}$
for $b = b'$
and $c = c'$
$\boxed{\boxed{b'}}, \boxed{\boxed{c'}}$
$e \in_R \{0, 1, 2, 3\}$
e
if $e = 0$
$\boxed{\boxed{b}} \leftrightharpoons \boxed{\boxed{c}}$
if successfull
keep $\boxed{\boxed{b'}}$ and $\boxed{\boxed{c'}}$
if $e = 1$
$\boxed{\boxed{b}} \leftrightharpoons \boxed{\boxed{c'}}$
if successful
keep $\boxed{\boxed{b'}}$ and $\boxed{\boxed{c}}$
if $e = 2$
$\boxed{\boxed{b'}} \leftrightharpoons \boxed{\boxed{c}}$
if successfull
keep $\boxed{\boxed{b}}$ and $\boxed{\boxed{c'}}$
if $e = 3$
$\boxed{\boxed{b'}} \leftrightharpoons \boxed{\boxed{c'}}$
if successfull
keep $\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$

Protocol 4.21: Non-destructive equality for Rudich's commitment

Here, in protocol 4.21, by the equation $\boxed{\boxed{b}} \leftrightharpoons \boxed{\boxed{c}}$ we mean that S proves to R using protocol 4.19 that the commitments $\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$ are equal. By "if succesfull keep $\boxed{\boxed{b'}}$ and $\boxed{\boxed{c'}}$" we mean that the new representations for $\boxed{\boxed{b}}$ and $\boxed{\boxed{c}}$ are respectively $\boxed{\boxed{b'}}$ and $\boxed{\boxed{c'}}$ as the other bit commitments might have been destroyed.

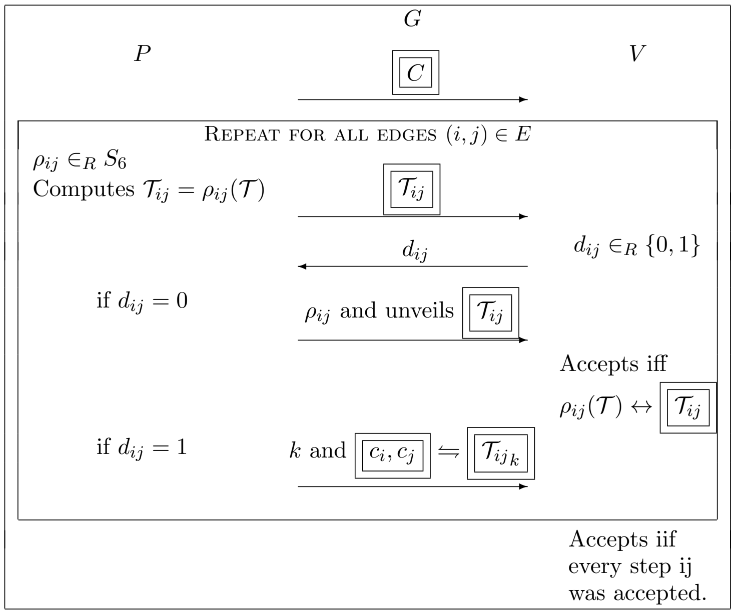
It is easy to observe that if S is honest, then S will pass this protocol with probability one and will still be comitted to values which are equal to the original values. Furthermore, the receiver will be *convinced* of their equality. But if S is cheating and $\boxed{\boxed{b}} \neq \boxed{\boxed{c}}$ then a finite case analysis proves that for at least two values of e, the prover cannot convince the verifier with probability one (remember that the the rudich's commitments always define a value). The soundness is at most one half. The protocol is completely symmetric and can easily be transformed into an unequality protocol with equivalent soundness. If we wish to have better soundness, we can use many $\boxed{\boxed{b}}$ and test them all[3].

ADD THE ANALYSIS THAT B=C but B'≠C'

4.6.2 Example

Let us use that equality protocol to do 3-Col again. No one would ever implement this protocol, but it cleanly uses together many concepts that we have learnt. We shall now use the symbol $\leftrightharpoons$ to mean equality as implemented by Protocol 4.21.

As an example, consider a new version or protocol 4.4. Let $\mathcal{T}$ be the table $\left[\begin{smallmatrix} R & R & G & G & B & B \\ G & B & R & B & G & R \end{smallmatrix}\right]^T$. Hence every row of $\mathcal{T}$ is a pair of colors which are different : (R, G) or (B, G) for example and each color is itself encoded using 2 bits.



G
P
V
$\boxed{\boxed{C}}$
Repeat for all edges $(i,j) \in E$
$\rho_{ij} \in_R S_6$
Computes $\mathcal{T}_{ij} = \rho_{ij}(\mathcal{T})$
$\boxed{\boxed{\mathcal{T}_{ij}}}$
d_{ij}
$d_{ij} \in_R \{0,1\}$
if $d_{ij} = 0$
ρ_{ij} and unveils $\boxed{\boxed{\mathcal{T}_{ij}}}$
Accepts iff
$\rho_{ij}(\mathcal{T}) \leftrightarrow \boxed{\boxed{\mathcal{T}_{ij}}}$
if $d_{ij} = 1$
k and $\boxed{\boxed{c_i, c_j}} \leftrightharpoons \boxed{\boxed{\mathcal{T}_{ij_k}}}$
Accepts iif every step ij was accepted.

Protocol 4.22: A Zero-Knowledge protocol for 3-Col

[3]the reader may notice that it is possible to commit to inconsistant values with some pairs XORing to zero and some pairs XORing to one. Such commitments will be discarded automatically if ever unveilled, and thus do not constitute any advantage over valid ones. See problem 4.12.

In Protocol 4.22 we use protocol 4.17 with table $\mathcal{T}$ and pairs of vertices color comming from the vector of color C in order to prove color difference between vertices. Here, the symbol $\boxed{a} \leftrightarrow b$ means that once unveiled, the commitment $\boxed{a}$ is accepted as a valid commitment to b. The prover first commits to a vector of colors C. Then for every edge (i, j) the prover commits to a row permutation of the table $\mathcal{T}$. Then either the provers opens that table to show that it is correctly constructed or the prover provides k, a row in the table $\mathcal{T}_{ij}$ and shows equality of the colors c_i and c_j with the colors contained in the table thereby proving that c_i is not the same color as c_j and this for every edge.

Completeness is obvious for any one who knows a coloring C for the graph G. Soundness is more tricky, but still a finite case analysis can be done. If two vertices have the same color in $\boxed{\boxed{C}}$, let us say vertex c_i is equal to vertex c_j, then the prover has to cheat at least once somewhere. They are two cases, either nor c_i nor c_j where involved in the previous steps of the protocol or one of them (possibly both) were involved in comparison with other edges c_l for some l.

Let us first assume none of them were involved in a previous steps. Then either the table $\mathcal{T}_{ij}$ to which P comits is valid or it is not. If it is not, then P gets caught with probability one half. If the table is valid, then for all rows in the table, there is at least one difference between (c_i, c_j) and ${\mathcal{T}_{ij}}_k$ and the prover will be caught with probability one fourth. So if c_i and c_j where not involved before, the prover can cheat with probability at most seven height.

If either one of is involved before, the cheater can try to change the future value of c_i for example. Assume c_i is compared to c_k before and that $c_i \neq c_k$. Then as protocol 4.21 allows, the prover can enter the protocol with two values that are distinct but submits new commitments to c_i' and c_k' which are not respectively equal to c_i and c_k. Then, as at least one bit of c_i' is different from its counterpart in c_i, then the prover might get caught with probability one quarter. Hence, if the Prover tries to change the value to which he is committed while still proving the unequality between $\boxed{\boxed{c_i}}$ and $\boxed{\boxed{c_k}}$, the prover will be caught with probability at least one height. If the prover succeeds in changing his values, then, down the road, he will not have to cheat anymore. Whatever happens, if C is not a valid coloring, the Prover needs to cheat at least once and will be caught doing so with probability at least one height. This probability is a constant, hence the full protocol 4.22 can be amplified by standard repetition.

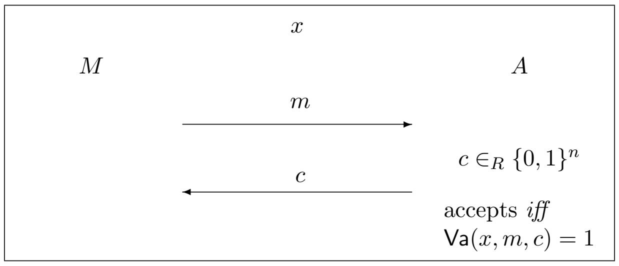
We shall not provide a full simulator but just remark that the simulator can cheat by choosing a table $\mathcal{T}'$ that is either equal to $\mathcal{T}$ or equal to $\mathcal{T}$ with one row replaced so that a relation between c_i and c_j can be proven (if both c_i and c_j are equal to R for example, then a row of the table is simply (RR)). Then, if that choice is equal to whatever V' wants to see, the simulator can complete simulation for the pair (c_i, c_j) of vertices; otherwise, he can just start over for that pair.

4.7 MA, AM and IP in ZK

In this section, we present a major result of complexity theory : PSPACE = ZKIP. We shall accomplish this using a result from chapter one : PSAPCE = IP = MA[*poly*], where *poly* is a polynomial in the size of the input. We shall do here only a sub-case, that is : MA[2] $\in$ ZKIP and leave the general result as an exercise.

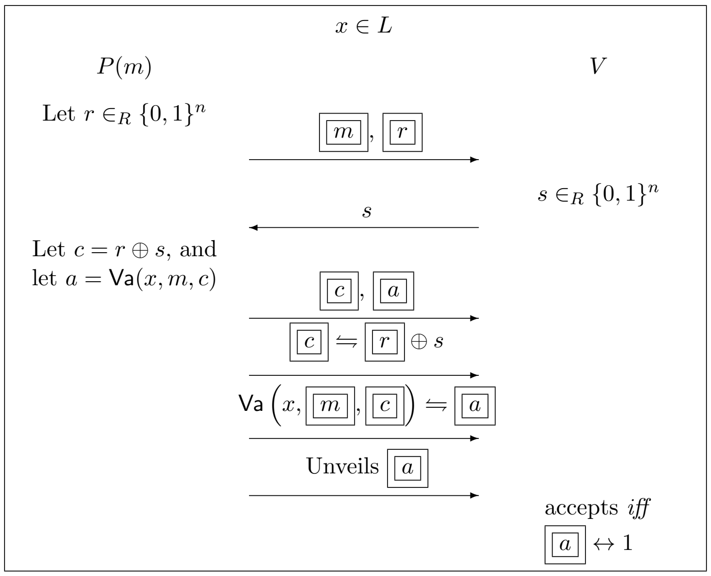
If a language $L \in$ MA, then there exists a poly-time Arthur that will accept value $x \in L$ with probability at least 2/3 when given an appropriate witness by Merlin, and that will reject value $x \notin L$ with probability at least 2/3 for whatever witness Merlin uses.

We shall assume that the provided bit commitment scheme is equipped with an equality protocol.We have a few things to prove.
Here is a general MA protocol proving that x belongs to a fixed language L:



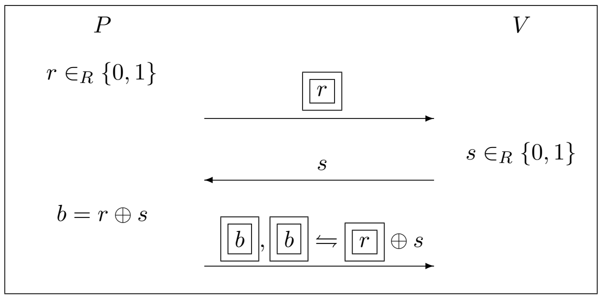
Protocol 4.23: Merlin-Arthur protocol

In Protocol 4.23, Va is a fixed deterministic validation circuit where the length (n) of the randomness c is polynomial in the length of the input $|x|$. Here is a zero-knowledge version of that protocol with some obvious details (by now, you should be comfortable with them) not shown.


$x \in L$
$P(m)$
V
Let $r \in_R \{0,1\}^n$
$\boxed{\boxed{m}}, \boxed{\boxed{r}}$
$s \in_R \{0,1\}^n$
s
Let $c = r \oplus s$, and
let $a = \mathsf{Va}(x, m, c)$
$\boxed{\boxed{c}}, \boxed{\boxed{a}}$
$\boxed{\boxed{c}} \leftrightharpoons \boxed{\boxed{r}} \oplus s$
$\mathsf{Va}\left(x, \boxed{\boxed{m}}, \boxed{\boxed{c}}\right) \leftrightharpoons \boxed{\boxed{a}}$
Unveils $\boxed{\boxed{a}}$
accepts *iff*
$\boxed{\boxed{a}} \leftrightarrow 1$

Protocol 4.24: Wrong ZK version of Merlin-Arthur protocol

Protocol 4.24 uses the following protocol named **coin tossing in a well** implicitly:

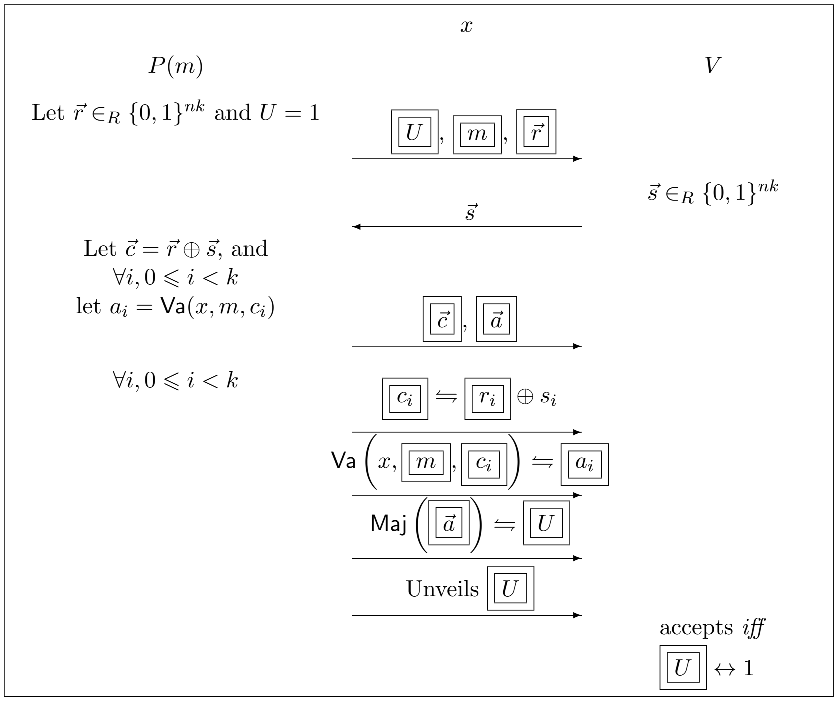

P
V
$r \in_R \{0,1\}$
$\boxed{\boxed{r}}$
s
$s \in_R \{0,1\}$
$b = r \oplus s$
$\boxed{\boxed{b}}, \boxed{\boxed{b}} \leftrightharpoons \boxed{\boxed{r}} \oplus s$

Protocol 4.25: Coin tossing in a well

In Protocol 4.25, the bit b is a fair coin if at least one of the parties is honest. Let V be honest, then whatever distribution used by P to choose its bit r, since the bit s is perfectly random, the bit b will also be. Note that if we use a perfectly concealing bit commitment scheme to commit to r, the choice of s cannot depend on the value of r. Also, since $\boxed{\boxed{r}}$ is transmitted before s is, it must be that r is independent of s. Hence, if P his honest, then the bit b will also be uniformly distributed. Hence the bit b is fair and unbiased. We qualify the primitive of "in a well" because the coin is tossed but one party does not know the outcome of the tossing. The analogy is that if you flip an actual coin in a well, you must get very close to the well to actually find out about the outcome.

Protocol 4.24 would definitely be an IP protocol and its completeness and soundness would be essentially the same as the MA protocol. We would need to amplify this protocol in order to get good soundness. But this would let V learn a vector of answers $\vec{a} = (a_0, a_1, \ldots, a_k)$. Although this looks harmless consider the following scenario. Let x have two witnesses and let the soundness vary with the witness. For example the protocol has soundness $3/4$ with one witness and $7/8$ with the second witness. After k repetitions, it would be possible with good probability (by the chernoff bound), to distinguish between the two witnesses. Intuitively, this is not a desirable property of a ZK protocol. In fact no simulator would be able to simulate that without the witness and without intimate knowledge of the witnesses of a given x. Remember, a simulator has to simulate for all members of the language.

This problem is solved by not revealing the individual a_i's as below:


x
$P(m)$
V
Let $\vec{r} \in_R \{0,1\}^{nk}$ and $U = 1$
$\boxed{\boxed{U}}, \boxed{\boxed{m}}, \boxed{\boxed{\vec{r}}}$
$\vec{s} \in_R \{0,1\}^{nk}$
$\vec{s}$
Let $\vec{c} = \vec{r} \oplus \vec{s}$, and
$\forall i, 0 \leqslant i < k$
let $a_i = \mathsf{Va}(x, m, c_i)$
$\boxed{\boxed{\vec{c}}}, \boxed{\boxed{\vec{a}}}$
$\forall i, 0 \leqslant i < k$
$\boxed{\boxed{c_i}} \leftrightharpoons \boxed{\boxed{r_i}} \oplus s_i$
$\mathsf{Va}\left(x, \boxed{\boxed{m}}, \boxed{\boxed{c_i}}\right) \leftrightharpoons \boxed{\boxed{a_i}}$
$\mathsf{Maj}\left(\boxed{\boxed{\vec{a}}}\right) \leftrightharpoons \boxed{\boxed{U}}$
Unveils $\boxed{\boxed{U}}$
accepts *iff*
$\boxed{\boxed{U}} \leftrightarrow 1$

Protocol 4.26: ZK version of Merlin-Arthur protocol

In this protocol, Maj is a Majority circuit. If $\boxed{\boxed{\vec{a}}}$ is a commitment to a string $(a_0, a_1, \ldots, a_{k-1})$, then $\mathsf{Maj}\left(\boxed{\boxed{\vec{a}}}\right) = \boxed{\boxed{1}}$ if and only if at least half the bits of $\vec{a}$ are ones. The strings $\vec{r}$ and $\vec{s}$ are parsed as k strings, r_i and s_i respectively, of n bits each.

In Protocol 4.26, the step $\mathsf{Va}\left(x, \boxed{\boxed{m}}, \boxed{\boxed{c_i}}\right) \leftrightharpoons \boxed{\boxed{a_i}}$ means that P and V evaluate gate by gate in zero-knowledge the circuit representing Va. Of course, P and V have to agree on a sequence of gates and their inputs, but this can be done before the protocol starts. We have seen in Section 4.6 how to compute a gate over commitments using a table; using the trick described above, this can even be done with random strings. Since we repeat this k times, this is in fact equivalent to running the circuit Va k times using k independent random strings. Since Protocol 4.23 was in MA, we know that Protocol 4.23 has soundness at most two thirds. Hence running this k time should accept $2k/3$ times on average and if k is large enough, the probability that more than half the runs do not accept is negligible (by the Chernoff bound). Note that the $\boxed{\boxed{a_i}}$ are never unveiled. Protocol 4.26 runs a majority circuit in zero-knowledge on the string $\vec{a}$ which is in fact the k results from the k invocations of Va. As just argued, this circuit should output 1 with overwhelming probability.

We can now argue that Protocol 4.26 is zero-knowledge.

Completeness:
If P is honest and receives a valid witness m, then P can follow the true protocol. For every i, the probability that $\mathsf{Va}(x, m, c_i) = 1$, is at least two thirds. Therefore, as argued before, with overwhelming probability the evaluation of the Maj circuit will yield a commitment to a 1 which once opened will still be a one. Therefore the Verifier will accept with overwhelming probability.

Soundness:
If the prover is cheating, then he needs to either cheat when he opens a commitment to the 1 at the end, cheat when he proves the equality between two values, or cheat when he computes a gate. He needs to cheat at least once. But since the equality procedure has an overwhelming probability of catching a cheating prover, this cannot be done reliably. If he tries to cheat a gate computation, he will get caught again with overwhelming probability as repeating k' times Protocol 4.17 ensures overwhelming soundness. Finally, if the prover could open a 1 whilst the commitment $\boxed{\boxed{U}}$ was in fact to a zero and this with non-negligible probability, this would contradict the fact that the bit commitment is binding. The compounded probability of cheating is bounded above by the sum, hence the protocol is sound.

Simulator: We shall give a more hand-wavy simulator this time as the Protocol is already high level. The Simulator will follow the protocol as P would, only with a bogus m as a witness. Since the bit-commitment is computationally concealing, this ensures the forged transcript is distributed in an undistinguishable way from the real one. The simulator will simulate honestly until the last gate of the Maj circuit. Let the last gate be $\boxed{\boxed{q}} \cdot \boxed{\boxed{t}} = \boxed{\boxed{U}}$. The simulator did commit to a one with $\boxed{\boxed{U}}$. The values q and t have been honestly computed as P would have. Hence until now, everything is distributed the same way. But since S has no witness, it must be, that with high probability, $q \cdot t \neq 1$ (note that if $q \cdot t = 1$, then the simulator does not even have to cheat and can in fact just finish the computation). Then S just needs to cheat this last gate. We know how to do that. Every turn, the simulator chooses to either commit to a valid table T or to a bad table T' and then respectively open T or prove equality between $(\boxed{\boxed{q}}, \boxed{\boxed{t}}, \boxed{\boxed{U}})$ and one row of the bad table T'. We have already seen that this protocol is zero-knowledge. As the bit-commitements are perfectly hiding, the forged transcript will be distributed just as the real transcript is, albeit proving something false this time. And then S just unveils $\boxed{\boxed{U}}$ which is an honest commitment to a one showing that A would accept M's proof.

Every single step of the protocol and of the simulation can be done in polynomial time. Hence this is a valid ZK protocol.

For languages L in AM we proceed exactly as above except that P and V jointly produce the random strings $\boxed{\boxed{c_i}}$ before P commits to his favorite witnesses $m_1, ..., m_k$. In the case of AM, Merlin is allowed to choose his witnesses as a function of $\boxed{\boxed{c_i}}$ (but not in MA). At the end of this process, V should believe that the committed output bit $\boxed{\boxed{a_i}}$ is indeed the output of $\mathsf{Va}\left(x, \boxed{\boxed{m_i}}, \boxed{\boxed{c_i}}\right)$. P unveils $\boxed{\boxed{U}} = \mathsf{Maj}\left(\boxed{\boxed{\vec{a}}}\right)$ and V accepts iff $\boxed{\boxed{U}} \leftrightarrow 1$.

Finally, for languages $L \in$ IP=PSPACE, the same technique is used on the AM[*poly*] protocol for L. We proceed exactly as above except that there are now polynomially many (ℓ) rounds of joint creation of randomness and commitments to witnesses. At the end of this process, V should believe that the committed output bit $\boxed{\boxed{a_i}}$ is indeed the output of $\mathsf{Va}\left(x, \boxed{\boxed{m_i^1}}, \boxed{\boxed{c_i^1}}, ..., \boxed{\boxed{m_i^\ell}}, \boxed{\boxed{c_i^\ell}}\right)$. P unveils $\boxed{\boxed{U}} = \mathsf{Maj}\left(\boxed{\boxed{\vec{a}}}\right)$ and V accepts iff $\boxed{\boxed{U}} \leftrightarrow 1$.

Problems

4.1 Prove that Protocol 4.3 is computationally conceiling. Assume that an adversary A to the concealing property of the bit-commitment scheme exists. Use this adversary as a prover to the GNI problem. You may asume A has a reset button.

4.2 Consider the following alternative simulator to 4.5 :

```
1. Initialize $V$:  copy fresh random bits to the verifier's random tape and
   fill-up the Auxiliary-Input tape.
2. Pick a random vector of colors $C''$.
3. Commit to that vector by giving $\boxed{C''}$ to $V$.
4. Wait for V to answer with an edge $(i,j)$
5. (a) if $c''_i = c''_j$, rewind to Step 2
   (b) if $c''_i \neq c''_j$, write transcript.
```

Simulator 4.27: Alternate Simulator for Graph 3-Coloring

Show that the above simulator succeeds with probability at least $2/3$ and thus the expected number of iterations before the simulator writes a transcript is only $3/2$.

4.3 Prove formally that Protocol 4.7 is computationally concealing.

4.4 Provide a computational zero-knowledge proof for the Hamiltonian circuit problem under suitable computational assumption. (A directed graph G is Hamiltonian if its edges contain a circuit visiting each vertex exactly one).

4.5 Prove that if a function f is injective, then the existence of a hard-core predicate of f implies that f is a one-way function. Start by proving that this does not have to hold for functions g which are not injective.

4.6 Prove equation (4.5).

4.7 Prove that Protocol 4.13 for equality is secure by reducing its security to the security of concealing property of the bit-commitment protocol.

4.8 Devise an computationally secure unbiased protocol to flip a random coin between two parties. Use this chapter's material.

4.9 Give a computational zero-knowledge interactive proof for the **Clique** problem under suitable computational assumption.

4.10 In Protocol 4.19 one pair of bit-commitments is destroyed, lets called it $\left(\boxed{\boxed{b_i}}, \boxed{\boxed{c_i}}\right)$. Show how the pair $\left(\boxed{\boxed{b_{1-i}}}, \boxed{\boxed{c_{1-i}}}\right)$ can still be useful and explain how one needs to modify larger protocols in order for your trick to work in them.

4.11 Provide a protocol akin to Protocol 4.21 but for inequality. Prove that it is sound.

4.12 Provide a simple protocol to autotest a Rudich's Commitment : a procedure by which a multiple Rudich's commitment via k pairs (supposed to all XOR to a single bit b) can be tested for concistancy, which succeeds only if the vast majority of the pairs really XOR to a single bit b. Make your procedure non-destructive as in Protocol 4.21.

4.13 In protocol 4.26, is it important that the verifier does not learn $r_i \oplus s_i$? Explain.

Problems

4.1 Prove that Protocol 4.3 is computationally conceiling. Assume that an adversary A to the concealing property of the bit-commitment scheme exists. Use this adversary as a prover to the GNI problem. You may asume A has a reset button.

4.2 Consider the following alternative simulator to 4.5 :

```
1. Initialize $V$:  copy fresh random bits to the verifier's random tape and
   fill-up the Auxiliary-Input tape.
2. Pick a random vector of colors $C''$.
3. Commit to that vector by giving $\boxed{C''}$ to $V$.
4. Wait for V to answer with an edge $(i,j)$
5. (a) if $c''_i = c''_j$, rewind to Step 2
   (b) if $c''_i \neq c''_j$, write transcript.
```

Simulator 4.28: Alternate Simulator for Graph 3-Coloring

Show that the above simulator succeeds with probability at least 2/3 and thus the expected number of iterations before the simulator writes a transcript is only 3/2.

4.3 Prove formally that Protocol 4.7 is computationally concealing.

4.4 Provide a computational zero-knowledge proof for the Hamiltonian circuit problem under suitable computational assumption. (A directed graph G is Hamiltonian if its edges contain a circuit visiting each vertex exactly one).

4.5 Prove that if a function f is injective, then the existence of a hard-core predicate of f implies that f is a one-way function. Start by proving that this does not have to hold for functions g which are not injective.

4.6 Prove equation (4.5).

4.7 Prove that Protocol 4.13 for equality is secure by reducing its security to the security of concealing property of the bit-commitment protocol.

4.8 Devise an computationally secure unbiased protocol to flip a random coin between two parties. Use this chapter's material.

4.9 Give a computational zero-knowledge interactive proof for the **Clique** problem under suitable computational assumption.

4.10 In Protocol 4.19 one pair of bit-commitments is destroyed, lets called it $\left(\boxed{\boxed{b_i}}, \boxed{\boxed{c_i}}\right)$. Show how the pair $\left(\boxed{\boxed{b_{1-i}}}, \boxed{\boxed{c_{1-i}}}\right)$ can still be useful and explain how one needs to modify larger protocols in order for your trick to work in them.

4.11 Provide a protocol akin to Protocol 4.21 but for inequality. Prove that it is sound.

4.12 Provide a simple protocol to autotest a Rudich's Commitment : a procedure by which a multiple Rudich's commitment via k pairs (supposed to all XOR to a single bit b) can be tested for concistancy, which succeeds only if the vast majority of the pairs really XOR to a single bit b. Make your procedure non-destructive as in Protocol 4.21.

4.13 In protocol 4.26, is it important that the verifier does not learn $r_i \oplus s_i$? Explain.

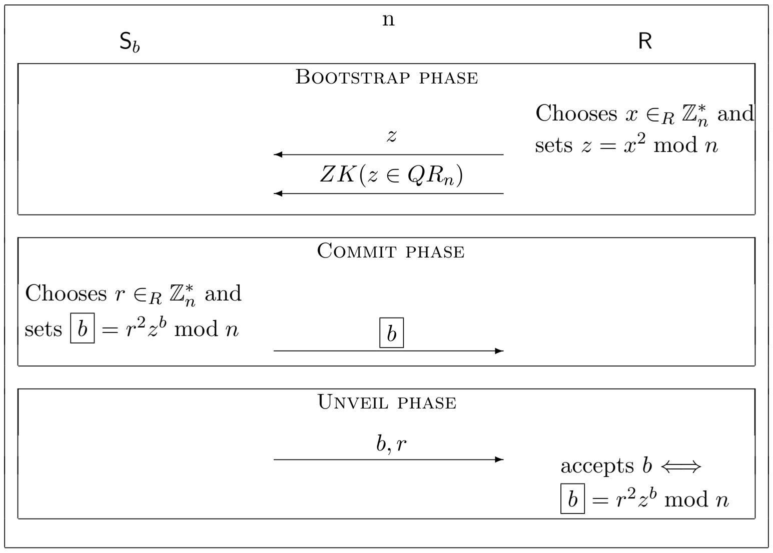
Chapter 5

Interactive Arguments

Until now, we have presented protocols that are secure even if the prover or the sender are infinitely powerful. However powerful is the prover, he cannot cheat and prove as isomorphic two graphs that are in fact not isomorphic. The prover could be lucky, but the probability that he is lucky cannot really be improved by any devious strategy. Hence the verifier is pretty convinced that the prover is not lying. In this chapter, we shall show how to build protocol which are secure against polynomial-time prover only. That is, any too powerful prover would be able to abuse the verifier and convince him of something false, i.e. break soundness. However, as we will see in this chapter, that this compromise allows to improve the zero-knowledge aspect and exhibit perfect zero-knowledge arguments for all languages in NP and even MA.

5.1 Perfectly Concealing Bit-Commitments

Upto this point, bit-commitments have been Perfectly, or statistically, binding for the sender and computationally concealing. That is for any prover, however powerful, the committed value cannot be changed when unveiling, whilst for all polynomial time verifiers, the committed value is unknown.

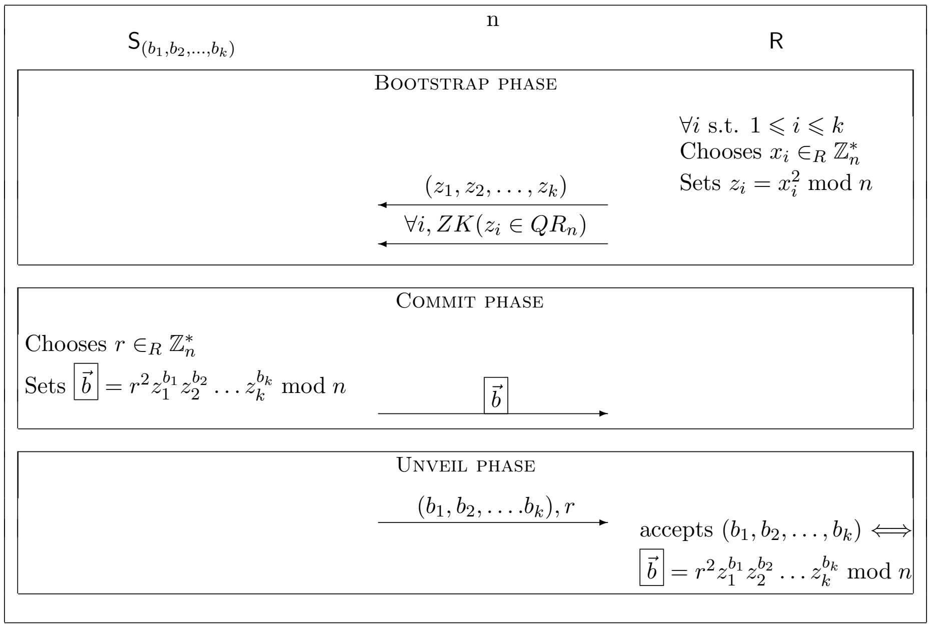

n
S_b
R
Bootstrap phase
Chooses $x \in_R \mathbb{Z}_n^*$ and sets $z = x^2 \bmod n$
z
$ZK(z \in QR_n)$
Commit phase
Chooses $r \in_R \mathbb{Z}_n^*$ and sets $\boxed{b} = r^2 z^b \bmod n$
$\boxed{b}$
Unveil phase
b, r
accepts $b \iff \boxed{b} = r^2 z^b \bmod n$

Protocol 5.1: Bit commitment based on Factoring

We shall now introduce a couple of commitment schemes which have the inverse properties: binding will be computational and concealing will be perfect or statistical. A first example was shown in Protocol 4.6 which is using Graph-isomorphism. As the sender is not aware of the permutation mapping G_0 to G_1 and should not be able to compute it in polynomial time, the protocol should be computationally binding. But as the two graphs are isomorphic (the zero-knowledge proof made sure of that), the receiver has no clue what the committed bit can be. Hence the protocol is statistically concealing. Protocol 5.1 is an example based on factoring.

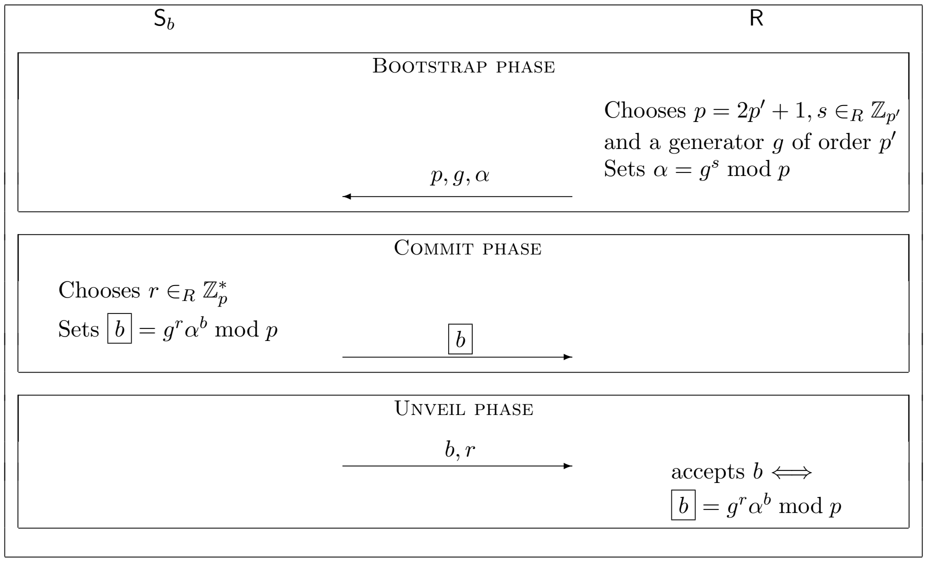
Observe that the bootstrap phase could also have a step to convince the Sender that n is an RSA integer, or has exactly two factors (see Exercise **??** in Chapter 4); this all depends on were n comes from. In this type of bit commitments, since the Receiver is choosing the public parameters, it is not in his interest to choose them weak. Observe a few things. First for all r there exists an r' such that $r^2 = {r'}^2 z \bmod n$, hence this protocol is perfectly concealing. Second, to cheat, S really needs to compute a square root $\bmod n$. Finally, it comes with equality. Let $\boxed{b} = r^2 z^b$ and $\boxed{c} = s^2 z^c$. Then $\boxed{b} \cdot \boxed{c} \triangleq (rs)^2 z^{b+c}$. Hence if $b = c = 0$, we can prove equality by returning $\sqrt{\boxed{b} \cdot \boxed{c}} = rs \bmod n$, if $b = c = 1$, we can prove equality by returning $\sqrt{\boxed{b} \cdot \boxed{c}} = rsz \bmod n$. Inequality can be proven by returning $\sqrt{z \cdot \boxed{b} \cdot \boxed{c}} = rsz \bmod n$.

If we wish for more efficiency, a parallel version can be used.


n
$\mathsf{S}_{(b_1,b_2,\dots,b_k)}$
R
Bootstrap phase
$\forall i$ s.t. $1 \leqslant i \leqslant k$
Chooses $x_i \in_R \mathbb{Z}_n^*$
Sets $z_i = x_i^2 \bmod n$
$(z_1, z_2, \dots, z_k)$
$\forall i, ZK(z_i \in QR_n)$
Commit phase
Chooses $r \in_R \mathbb{Z}_n^*$
Sets $\boxed{\vec{b}} = r^2 z_1^{b_1} z_2^{b_2} \dots z_k^{b_k} \bmod n$
$\boxed{\vec{b}}$
Unveil phase
$(b_1, b_2, \dots . b_k), r$
accepts $(b_1, b_2, \dots, b_k) \Longleftrightarrow$
$\boxed{\vec{b}} = r^2 z_1^{b_1} z_2^{b_2} \dots z_k^{b_k} \bmod n$

Protocol 5.2: Parallel bit commitment based on Factoring

Let us present a last example based on the discrete logarithm problem. Let p be a Sophie Germain prime (strong prime) and let g be a generator of $\mathbb{Z}_p^*$ and α be equal to $g^s \bmod p$ for some s.


S_b
R
BOOTSTRAP PHASE
Chooses $p = 2p' + 1, s \in_R \mathbb{Z}_{p'}$
and a generator g of order p'
Sets $\alpha = g^s \bmod p$
p, g, α
COMMIT PHASE
Chooses $r \in_R \mathbb{Z}_p^*$
Sets $\boxed{b} = g^r \alpha^b \bmod p$
$\boxed{b}$
UNVEIL PHASE
b, r
accepts $b \iff$
$\boxed{b} = g^r \alpha^b \bmod p$

Protocol 5.3: Bit commitment based on Discrete Logarithm

Now Protocol 5.3 is obviously concealing, since $\boxed{b}$ is just a random element in $\mathbb{Z}_p^*$ which distribution does not depend on whether b is zero or one. As for binding, consider what is needed to unveil both ways, values $\boxed{b} \equiv g^{r_0}\alpha^0 \bmod p$ and $\boxed{b} \equiv g^{r_1}\alpha^1 \bmod p$, then $\alpha \equiv g^{r_0 - r_1} \bmod p$, hence cheating is akin to computing the discrete logarithm of α. Note that this protocol can easily be made parallel.

5.2 Zero-Knowledge Interactive Arguments

What would happens if we were to use a perfectly concealing bit-commitment scheme in a Zero-Knowledge protocol for 3-Col for example. Obviously, the protocol would not be sound anymore as a very powerful prover could cheat and change his commitments at will. But if we were to define interactive proofs for polynomial-time provers only, we would still get something convincing, not as convincing as interactive proofs but still convincing. Here is the definition of **Interactive Arguments** or also sometimes called **computationally sound interactive proofs**:

Definition 13 (Interactive Argument) *A pair of Turing machines P and V, both probabilistic-polynomial time, constitute an Interactive Argument system for the language $\mathcal{L}$ if the interaction between the two machines has the two following properties:*

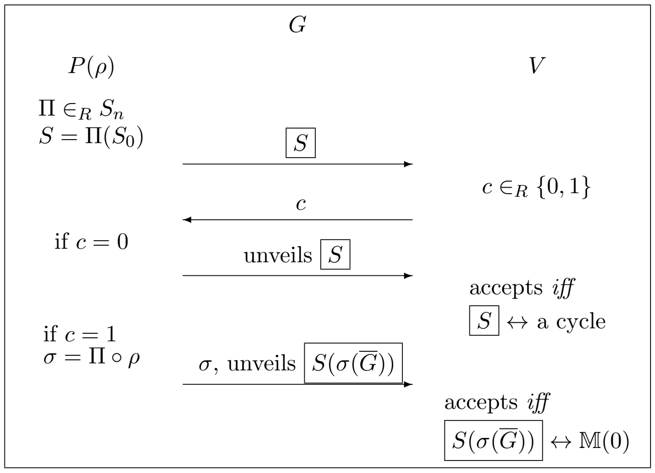
***Completeness**: for all $x \in \mathcal{L}$*

$$\Pr[V \textit{ accepts } x \textit{ after interacting with } P] \geqslant \frac{2}{3}; \tag{5.1}$$

***Soundness**: if for all $x \notin \mathcal{L}$ and all PPT P'*

$$\Pr[V \textit{ accepts } x \textit{ after interacting with } P'] \leqslant \frac{1}{3}. \tag{5.2}$$

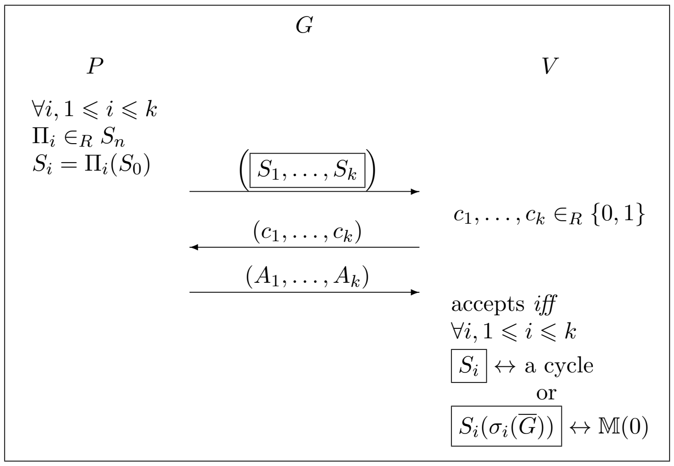
Here is an interactive Argument protocol for the Hamiltonian cycle problem.


G
$P(\rho)$
V
$\Pi \in_R S_n$
$S = \Pi(S_0)$
$\boxed{S}$
$c \in_R \{0,1\}$
c
if $c = 0$
unveils $\boxed{S}$
accepts *iff*
$\boxed{S} \leftrightarrow$ a cycle
if $c = 1$
$\sigma = \Pi \circ \rho$
σ, unveils $\boxed{S(\sigma(\overline{G}))}$
accepts *iff*
$\boxed{S(\sigma(\overline{G}))} \leftrightarrow \mathbb{M}(0)$

Protocol 5.4: A Zero-Knowledge protocol for Hamiltonian cycle

Here, S_0 is a fixed adjacency matrix with a fixed cycle and no other edge; that matrix is also known from V. The permutation ρ maps the hamiltonian cycle contained in G to the one contained in S_0; which means that σ maps the Hamiltonian cycle in G to the one in S. The notation $S(\sigma(\overline{G}))$ means that only the entries which encodes non-edges of $\sigma(G)$, which we denote by $\sigma(\overline{G})$. Of course a non-edge is a pair of vertex (i, j) such that $(i, j) \notin E$. Finally, $\mathbb{M}(0)$ is the zero matrix.[1] Therefore "unveils $\boxed{S(\sigma(\overline{G}))}$" means that for all non-edges (i, j) of $\sigma(G)$, the prover opens potential edge (i, j) in $\boxed{S}$ and the verifier accepts if and only if (i, j) is not an edge in S.

Could we do parallel repetition of the previous protocol? Let's try the obvious thing.


G
P
V
$\forall i, 1 \leqslant i \leqslant k$
$\Pi_i \in_R S_n$
$S_i = \Pi_i(S_0)$
$\left(\boxed{S_1, \ldots, S_k}\right)$
$c_1, \ldots, c_k \in_R \{0,1\}$
$(c_1, \ldots, c_k)$
$(A_1, \ldots, A_k)$
accepts *iff*
$\forall i, 1 \leqslant i \leqslant k$
$\boxed{S_i} \leftrightarrow$ a cycle
or
$\boxed{S_i(\sigma_i(\overline{G}))} \leftrightarrow \mathbb{M}(0)$

Protocol 5.5: A parallel Interactive Argument for Hamiltonian cycle

In protocol 5.5, A_i is the answer to challenge c_i: that is, either A_i is "unveils $\boxed{S_i}$" if c_i is zero or A_i is the pair "σ_i, unveils $\boxed{S_i(\sigma_i(\overline{G}))}$" if $c_i = 1$ and $\sigma_i = \Pi_i \circ \rho$. Protocol 5.5 is three messages constitued each of a k-tuple whilst protocol 5.4 is three mesages each of only one singlet and it has to be repeated k times in order to get negligeable soundness. Hence the parallel version of

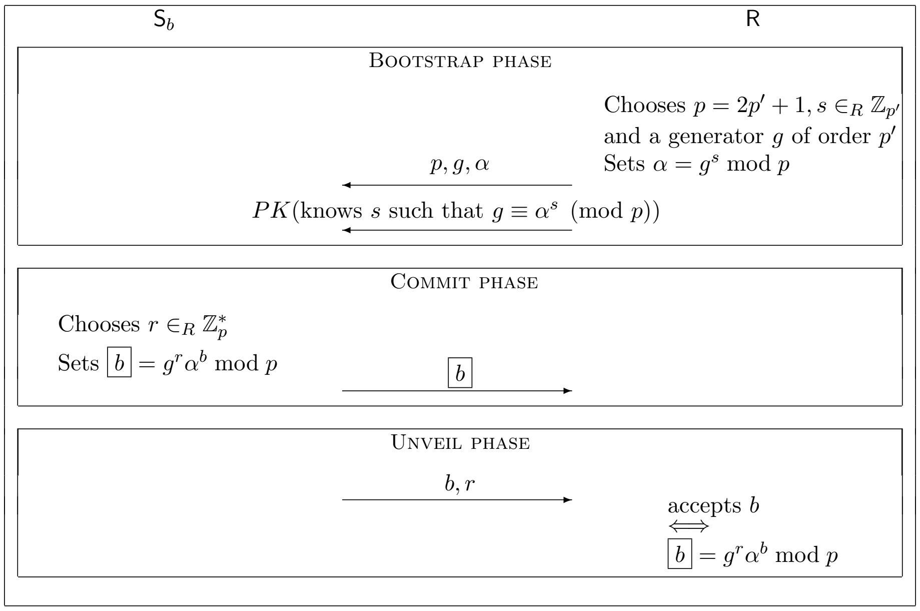
[1] See problem 5.4

the interactive argument is much more efficient in the number of rounds, and thus, the number of messages while keeping the computation requirement at the same complexity.

As for the zero-knowledge property of protocol 5.5 there is a catch. We can easily build a simulator for protocol 5.5 using the standard trick of just hoping that the verifier chooses the question for which the simulator prepared an answer. But for protocol 5.5 this cannot work as the probability that the verifier asks the question vector for which the simulator prepared an answer is negligible. Even worse, the verifier could use a cryptographic hash function h to hash the vector $\left(\boxed{S_0}, \ldots \boxed{S_k}\right)$ to a vector of bits $(c_1, \ldots, c_k)$. This is no proof that we cannot simulate protocol 5.5 but it it certainly good intuition to why it might be hard or impossible.

There are two easy solutions to this conundrum.

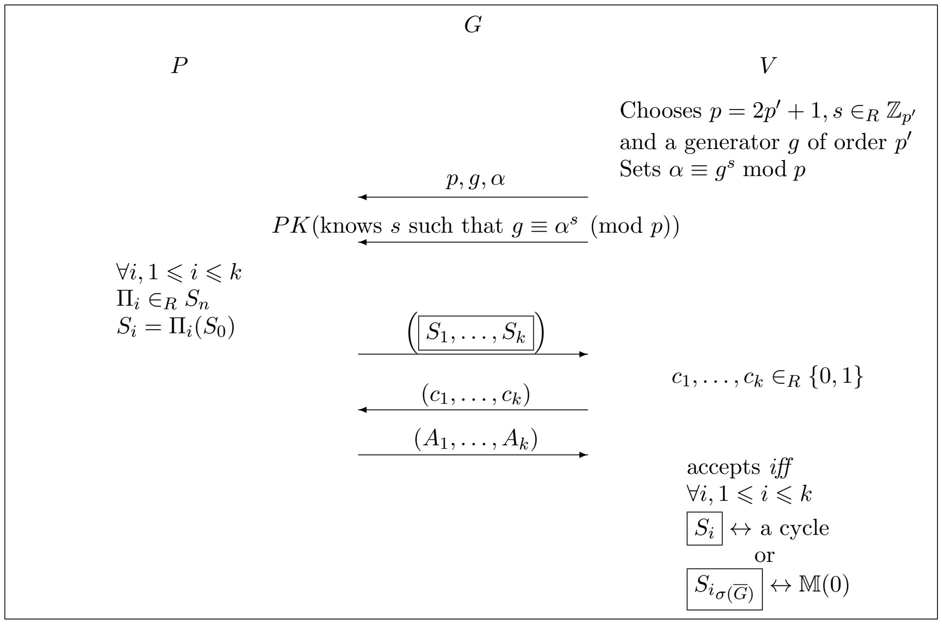
A first solution is to modify the bit commitment: use a perfectly concealing bit commitment with a twist. Let us revisit the bootstrap phase of protocol 5.3.


S_b
R
BOOTSTRAP PHASE
Chooses $p = 2p' + 1, s \in_R \mathbb{Z}_{p'}$
and a generator g of order p'
Sets $\alpha = g^s \bmod p$
p, g, α
$PK(\text{knows } s \text{ such that } g \equiv \alpha^s \pmod p)$
COMMIT PHASE
Chooses $r \in_R \mathbb{Z}_p^*$
Sets $\boxed{b} = g^r \alpha^b \bmod p$
$\boxed{b}$
UNVEIL PHASE
b, r
accepts b
$\Longleftrightarrow$
$\boxed{b} = g^r \alpha^b \bmod p$

Protocol 5.6: Bit commitment based on Discrete Logarithm with a twist

In this protocol, the fact that the Receiver proves that he knows s such that $g \equiv \alpha^s \pmod p$ brings nothing to the sender in the Interactive Argument aspects, as long as the proof of knowledge is sufficiently discreet. As discussed earlier, zero-knowledge is not necessary for this proof of knowledge. Indeed what matters is the fact that it does not reduce soudness of the main Interactive Argument. To this effect, the notion of *witness hiding* is sufficient. But for the simulator who needs to simulate a zero-knowledge argument, this little addition has a huge impact. The simulator can simply run the extractor on R to retrieve s. Once the simulator knows s, he can cheat (break binding of) the commitments at will.

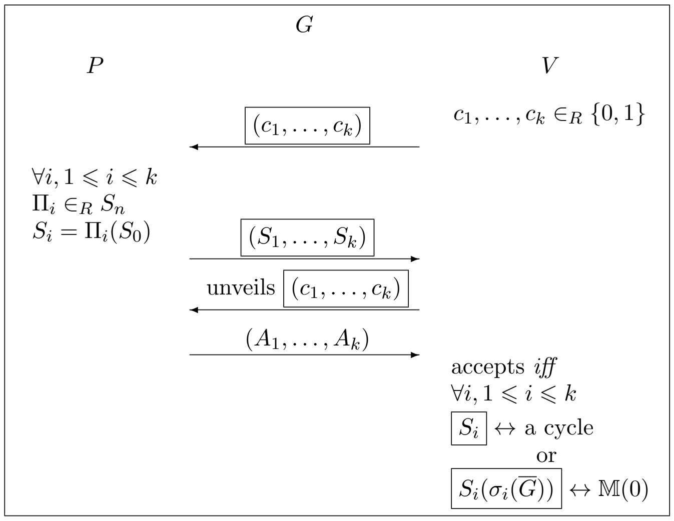
Using this new protocol we can revisit protocol 5.5


G
P
V
Chooses $p = 2p' + 1, s \in_R \mathbb{Z}_{p'}$
and a generator g of order p'
Sets $\alpha \equiv g^s \bmod p$
p, g, α
PK(knows s such that $g \equiv \alpha^s \pmod p$)
$\forall i, 1 \leqslant i \leqslant k$
$\Pi_i \in_R S_n$
$S_i = \Pi_i(S_0)$
$\left(\boxed{S_1, \ldots, S_k}\right)$
$c_1, \ldots, c_k \in_R \{0, 1\}$
$(c_1, \ldots, c_k)$
$(A_1, \ldots, A_k)$
accepts *iff*
$\forall i, 1 \leqslant i \leqslant k$
$\boxed{S_i} \leftrightarrow$ a cycle
or
$\boxed{S_{i_{\sigma(\overline{G})}}} \leftrightarrow \mathbb{M}(0)$

Protocol 5.7: A parallel argument for Hamiltonian cycle

The simulator for this protocol is very straightforward. But the most remarkable thing is that this proof has negligeable soundness using only 7 messages (the proof of knowledge can be run in parrallel using three messages). However, in order to prove formally that the parallel proof of knowledge does not weaken the soundness condition of the Interactive Argument, a formal definition of Witness Hiding would be necessary[2]. We choose not to go that way.

As a second approach, we can change the protocol and force V to commit first to his challenges before the Prover commits to the S_i's. This is very natural strategy that has been extensively used to parallelize sequential protocols.


G
P
V
$\boxed{(c_1, \ldots, c_k)}$
$c_1, \ldots, c_k \in_R \{0, 1\}$
$\forall i, 1 \leqslant i \leqslant k$
$\Pi_i \in_R S_n$
$S_i = \Pi_i(S_0)$
$\boxed{(S_1, \ldots, S_k)}$
unveils $\boxed{(c_1, \ldots, c_k)}$
$(A_1, \ldots, A_k)$
accepts *iff*
$\forall i, 1 \leqslant i \leqslant k$
$\boxed{S_i} \leftrightarrow$ a cycle
or
$\boxed{S_i(\sigma_i(\overline{G}))} \leftrightarrow \mathbb{M}(0)$

Protocol 5.8: A parallel Zero-Knowledge protocol for Hamiltonian cycle

[2]Do not be mislead thinking that the situation is exactly the same as in Protocol 3.7 where a similar reasoning was used. It is a lot more subtle here than back then.

This strategy, protocol 5.8, works for Interactive Arguments as well as for Interactive Proofs, see Problem 5.6, and the simulator is fairly straight forward. However, the Bit Commitment used by the Receiver should be of the opposite type as the one used by the Sender so to keep computational assumptions all on the same side. For instance, If the Prover uses Protocol 5.3 for his commitments then the Verifier uses Protocol 4.9 for his commitments. If we did not follow these guidelines, the resulting protocol would neither be perfect Zero-Knowledge nor an Interactive Proof: it would be a Computational Zero-Knowledge Interactive Argument, an unnecessary compromise. Note that the same simulator will work for both Interactive Arguments and for Interactive Proofs but the proof of expected polynomial running in each case is rather different...

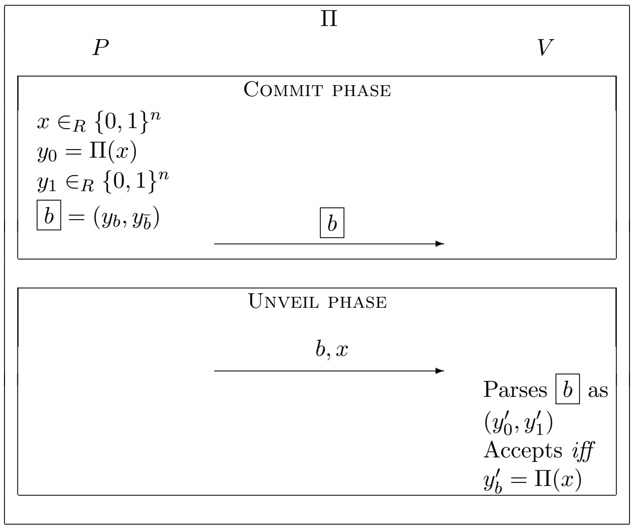
Since we know in general that bit commitments that are computationally concealing can be built from any one-way function, a similar result for bit commitments that are computationally binding would be desirable. This will be the topic of the next Section.

5.2.1 Arguments vs Proofs

5.3 Bit commitments based on any one-way hash function

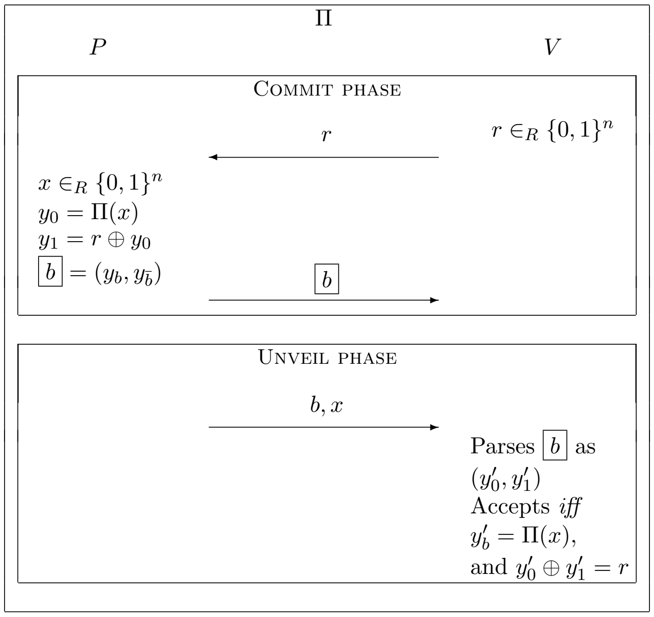
5.4 Bit commitments based on any one-way permutation

Can we build a general perfectly concealing bit-commitement using an arbitrary one-way permutation ? Let us try a few things so to build intuition of the final solution. Let P and V be probabilistic-polynomial time Turing machines and let Π be a one-way permutation over n-bit strings.



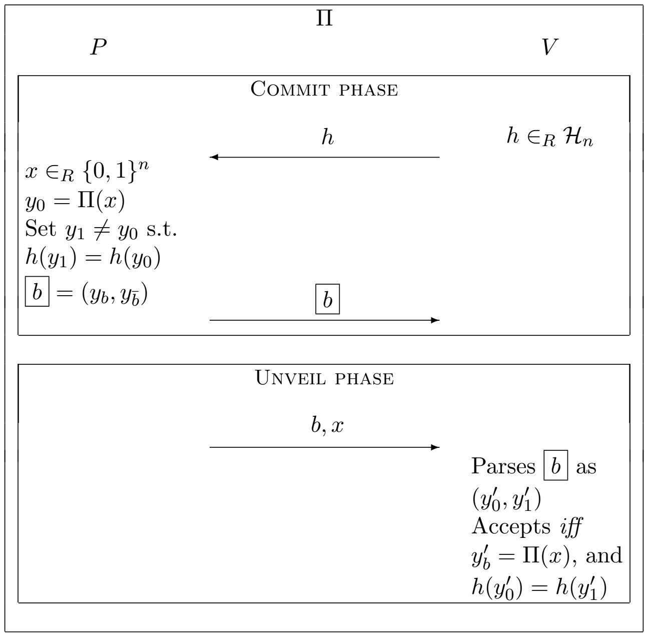
Protocol 5.9: A first attemp at BC using One-Way permutations

It is not hard to see that this protocol is perfectly concealing but perfectly useless as it is very easy for P to unveil either ways, as P can choose y_1 maliciously as the image under Π of another known x'. We need to force the value y_1 on P somehow as a function of y_0. Let's add the condition $r = y_0 \oplus y_1$ for some randomly selected r.



Protocol 5.10: A second attempt at BC using One-way permutation

To cheat, the prover needs to find x and x' such that $\Pi(x') \oplus \Pi(x) = r$. At first sight, that seems hard to solve, but at the same time it does not seem equivalent to inverting the one-way permutation on a given input. It would be a non-trivial supplemental assumption on the permutation Π. But hash functions have properties that could do it. Let $\mathcal{H}_n : \{0,1\}^n \longrightarrow \{0,1\}^{n-1}$ be a hashing function familly. Let's try to include that in our protocol.


Π
P
V
Commit phase
h
$h \in_R \mathcal{H}_n$
$x \in_R \{0,1\}^n$
$y_0 = \Pi(x)$
Set $y_1 \neq y_0$ s.t. $h(y_1) = h(y_0)$
$\boxed{b} = (y_b, y_{\bar{b}})$
$\boxed{b}$
Unveil phase
b, x
Parses $\boxed{b}$ as (y'_0, y'_1)
Accepts *iff* $y'_b = \Pi(x)$, and $h(y'_0) = h(y'_1)$

Protocol 5.11: A second BC using One-way permutation

As is, this protocol is not very secure as we have the same problem as in protocol 5.10. That is given h can the prover find x_0 and x_1 such that $y_0 = \Pi(x_0)$, $y_1 = \Pi(x_1)$ and $h(y_0) = h(y_1)$. This is not known to be equivalent to inverting Π on a random point. Indeed, Protocol 5.10 is a special case of Protocol 5.11, see exercice **??**. Fortunately, there is a solution to this problem called "interactive hashing".

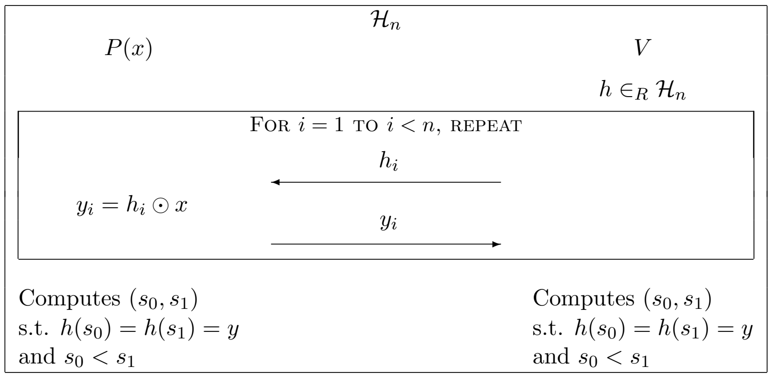
Let the familly $\mathcal{H}_n$ be the set of maximum rank $n-1 \times n$ binary matrices. Then for a random matrix M in $\mathcal{H}_n$, we have that $h(x) \triangleq Mx$, where x, and the h_i's below, are interpreted as n-bit column vectors. Or if

$$M = \begin{pmatrix} h_1^T \\ h_2^T \\ \dots \\ h_{n-1}^T \end{pmatrix}$$

then

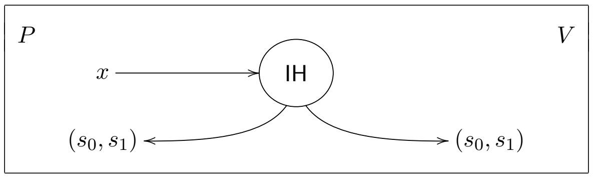
$$h(x) = \sum_{i=1}^{n-1} (h_i^T \odot x) e_i \tag{5.3}$$

where the symbol $\odot$ means "binary dot product" and the e_i's are the canonical base vectors $(1, 0, 0, ..., 0), (0, 1, 0, ..., 0), ..., (0, 0, 0, ..., 1)$. Interactive hashing is simply a protocol to evaluate equation 5.3 bit by bit. The sender sends the matrix M row by row to the receiver, who reponds with one bit of $h(x)$. The net effect, is that the Sender cannot choose both x_0 and x_1 together that would allow him to cheat as the Sender does not know yet the full function h. In the next protocol, Protocol 5.12, the y_i are the bits of vector $y = h(x)$.


$\mathcal{H}_n$
$P(x)$
V
$h \in_R \mathcal{H}_n$
FOR $i = 1$ TO $i < n$, REPEAT
h_i
$y_i = h_i \odot x$
y_i
Computes (s_0, s_1) s.t. $h(s_0) = h(s_1) = y$ and $s_0 < s_1$
Computes (s_0, s_1) s.t. $h(s_0) = h(s_1) = y$ and $s_0 < s_1$

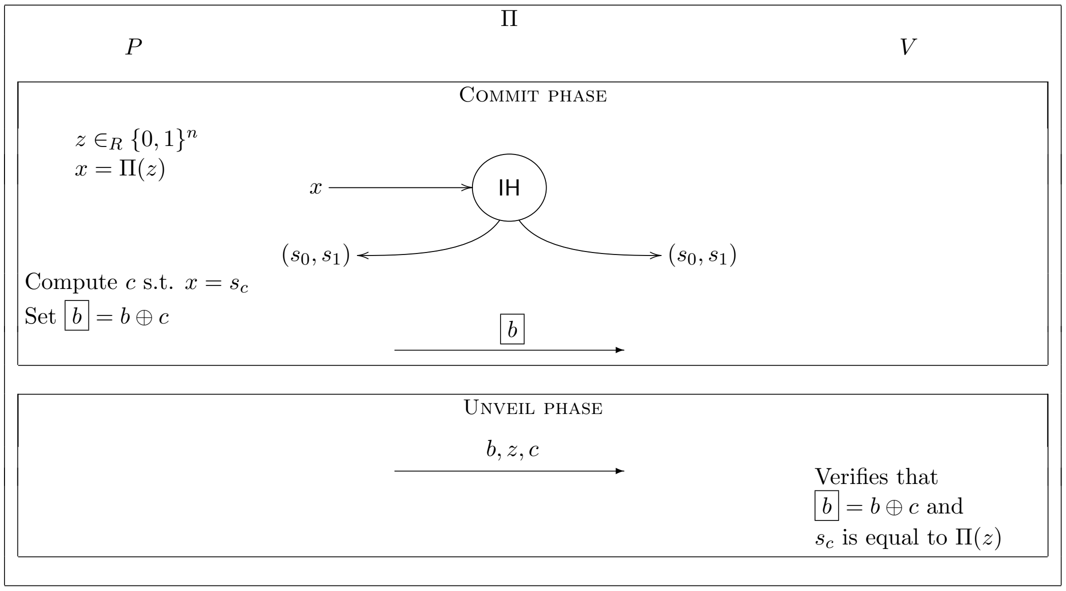
Protocol 5.12: Interactive Hashing

We can schematise the last protocol using this notation



Protocol 5.13: Interactive Hashing, notation

In protocol 5.13, both P and V know s_0 and s_1 but only the prover knows x, or only the prover knows c such that $x = s_c$. From V's perspective, that the prover uses s_0 or s_1 as input to the protocol, every single step of protocol 5.12 would yield exactly the same messages. Hence, this protocol is already concealing in some sense. Using this and a one-way permutation, we can build a perfectly-concealing computationally binding bit-commitment.



Protocol 5.14: BC using One-way permutation

As c is totally unknown to V, b is perfectly concealed in $\boxed{b}$; this is just an instance of one-time pad. What can be shown is that if there exists a P' that can open both 0 and 1 with good probability, then we can build an inverter to the one-way permutation Π.

First thing first, by good probability of opening 0 or 1 we mean that after the commit phase is done we have

$$\Pr[\text{unveils a zero \& succeeds}] + \Pr[\text{unveils a one \& succeeds}] \geqslant 1 + \frac{1}{poly(|z|)}$$

Here is the proof idea using : construct an inverter I for Π on a given value ν using a cheating P' as a black-box. The following inverter nearly does that. In reality, to make this argument work, one needs a very careful analysis as there are pitfalls.

```
1. Copy fresh random bits to $P'$'s random tape.
2. For $i = 1$ to $n - 1$ do
   (a) choose $h_i \in_R \{0,1\}^{n-1}$ s.t. for all $j < i$ $h_i \perp h_j$.
   (b) Hand $h_i$ to $P'$ and wait for $y_i$.
   (c) i. if $y_i = h_i \odot \nu$ then $i++$
       ii. else rewind $P'$ and try again for the same $i$
3. Wait for $\boxed{b}$.
4. Ask for unveiling and wait for $b, z, c$
5. if $\Pi(z) = \nu$, then output $z$ else output $\perp$.
```

Inverter 5.15: Inverter for Π

As P' is a good cheater, the probability that P' outputs a value z which belongs to $\Pi^{-1}(\nu)$ and thus that I can invert ν is at least $1/poly(|z|)$.

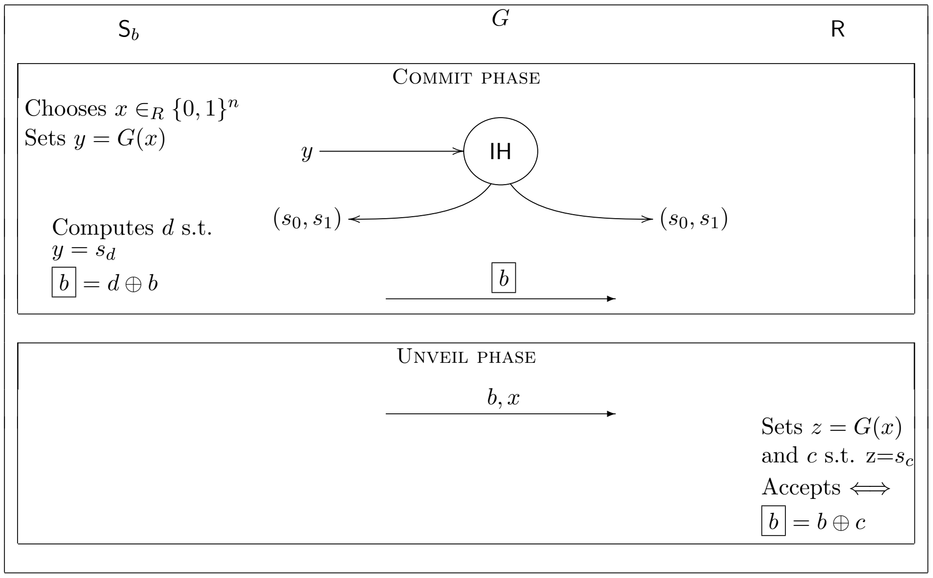
5.4.1 Extension to One-Way Functions

5.4.2 Interactive Hashing application

We shall now present another protocol for bit commitment using interactive hashing, but we shall use a different property of interactive hashing to show that the bit-commitment scheme is statistically binding but computationally concealing. We shall use the following theorem:

Theorem 6 (Savvides Theorem) *Let E, a subset of $\{0,1\}^n$, be a set of* good *strings. If the sender and the receiver enter an Interactive Hashing protocol where the receiver behaves honestly, then the probability that the second string (the one not chosen at the onset of the protocol by the sender) belongs to E is at most $16|E|/2^n$.*

The notion of a *good* string is application dependant (Usually consists of a set of strings which help the sender to cheat). In our case it will be a string which is in the co-domain of pseudo-random generator. So if $|E|/2^n$ is negligible, then the probability the sender can force both strings, s_0 and s_1, to be in the set E is negligible. Let G be a pseudo-random generator (we know those can be build from any one-way function) mapping n bit strings to $(1+\delta)n$ bit strings. The following is a commitment scheme:



Protocol 5.16: A commitment scheme based on pseudo random generator and interactive hashing

In the commit phase, the sender first choses a string x and computes its image y under the pseudo-random generator G. The Sender then enters the interactive hashing protocol with this y and gets a y' out of the interactive hashing. The receivers gets the pair of string (s_0, s_1) out of the interactive hashing protocol and by the virtue of this protocol, there exists d such that $y = s_d$. Then to commit, the sender computes d which is simply a bit saying wether y' is smaller than y or not and uses this bit to encrypt the bit b. This encryption is the commitment to b.

To unveil, the senders sends b and x. From x, the receiver can compute y and verify that $\boxed{b}$ was indeed a valid commitment to b using s_0 and s_1.

Concealing

With high probability, if the receiver is honest, y will be in the range of G and y' will not.By the properties of G, the output of G cannot be distinguished from the uniform distribution by any PPT distinguisher. Let $\lambda(G(X))$ be the distribution of the output of $G(X)$ in the range. That is for all strings not in the image but in the range, the probability is zero, and for all the others, the probability is a function of $\lambda(X)$, where X is the distribution of seed to G. To compute d, the receiver has to be able to distinguish between $(\lambda(G(X)), \mathbb{U}_{(1+\delta)n})$ and $(\mathbb{U}_{(1+\delta)n}, \lambda(G(X)))$. But we know that $(\lambda(G(X)), \mathbb{U}_{(1+\delta)n})$ is computationally indistinguishable form $(\mathbb{U}_{(1+\delta)n}, \mathbb{U}_{(1+\delta)n})$ and that $(\mathbb{U}_{(1+\delta)n}, \mathbb{U}_{(1+\delta)n})$ is computationally indistinguishable from $(\mathbb{U}_{(1+\delta)n}, \lambda(G(X)))$. As computational indistinguishability obeys the triangle inequality, we conclude that the receiver cannot compute b and hence that the bit-commitment scheme is computationally concealing.

Binding

Let us assume that if the second string generated by the interactive hashing, that is y', is also in the range of G, then S can always cheat (That might not be the case, but in the worst case it would be possible). The range size is $2^{n(1+\delta)}$. The size of the domain of G (which is the size of the good string set) is 2^n. Therefore by Savvides' theorem, we know that the probability that the interactive hashing protocol outputs a string which has a pre-image under G is bounded above by $16 \cdot 2^n/2^{n(1+\delta)}$ which is equal to $16/2^{\delta n}$. Since δ is a constant, then this probability is negligible for large enough n.

Note that this protocol is a hybrid between two protocols : The bit commitment based on pseudo-random generator, Protocol 4.12, and the bit-commitment based on one-way permutation and interactive hashing, Protocol 5.14 . The security for binding is based on a combinatoric (or information theoretic) argument just as in the protocol based on pseudo-random generator — but it uses a strong result for interactive hashing allowing it to use a much smaller expansion factor; the security for concealing is akin to the proof for concealing for the protocol based pseudo-random generator. Whilst the protocol itself is almost identical to the bit-commitment protocol based on one-way permutation and interactive-hashing: the only difference is that the one-way permutation was replaced by a pseudo-random generator (which can be build from any one-way function). This protocol requires a smaller expansion factor to reach security than Protocol 4.12 at the cost of more interaction.

5.4.3 Bounded round Interactive Hashing

5.5 MA in Perfect Zero-Knowledge

Problems

5.1 Why is concealing statistical in Protocol 5.1. Hint: Think of Protocol 3.7.

5.2 Prove that the equality and inequality procedures for Protocol 5.1 are secure in all aspects.

5.3 What would happen in Protocol 5.2 if only one z_i was exchanged and yet S and R were still trying to run a parallel version of the protocol?

5.4 Prove that protocol 5.4 is an interactive zero-knowledge argument. Provide proofs for soundness and completeness and a simulator.

5.5 Provide a proof of knowledge and extractor for the bootstrap phase of protocol 5.6.

5.6 Right after Protocol 5.8 it is said that this solution works with both interactive proofs and interactive arguments. There are differences on how to apply this technique in both cases. Can you find one? Explain why interactive proofs are more *challenging* than interactive arguements.

5.7 Show that the proof of knowledge of protocol 5.7 can be run in parallel and thus needs only three messages. Also prove that it does not bias the soundness of Protocol 5.7

5.8 Provide the full simulator for the argument of protocol 5.7.

5.9 Let X and Y be random variables over $\{0,1\}^n$, prove the following equation which is used implicitly in the proof of binding of protocol 5.16 :

$$D(X\mathbb{U}, Y\mathbb{U}) = D(X, Y).$$

5.10 Let X, Y and Z be random variables over $\{0,1\}^n$, prove the triangle inequality for statistical distance that is :

$$D(X, Z) \leqslant D(X, Y) + D(Y, Z)$$

and that the equation is saturated for independent variables.

Chapter 6

Oblivious Transfer

Up until now we have developed protocols that allow the sender S, which has input x and the receiver R to compute a function f over x and both parties receive the answer, that is $f(x)$. In this chapter, we shall develop another primitive, called **Oblivious Transfer** which will allow S, who possess x, and R, who possess y, to compute $f(x, y)$ and R receives the answer while keeping the information as low as possible on the other players input given the output of the function f.

Oblivious transfers comes in multiple flavors, two of which we shall explore here. We shall start with the Oblivious Transfer of Rabin.

6.1 Rabin Oblivious Transfer

The Rabin Oblivious Transfer is akin to an erasure channel for which the receiver either receives a bit and knows that the bit was valid (was not modified by the channel), or he receives an erasure symbol and hence is clueless about the value input into the channel by the sender.

Schematicaly we shall denote such an Oblivious Transfer as follows :

$$\begin{tikzcd}
S & & R \\
b_s \arrow[rr, dashed, "\epsilon"'] & & b_r
\end{tikzcd}$$

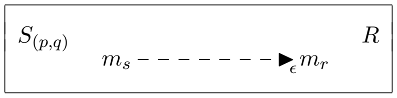
Protocol 6.1: Rabin Oblivious Transfer

And the channel has the following properties :

- $\Pr[b_r = b_s] = \epsilon$, for both values of b_s.
- $\Pr[b_r = \perp] = 1 - \epsilon$

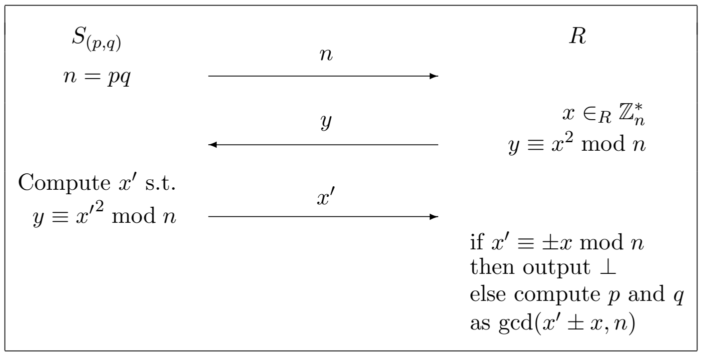
And finally, the sender S has no idea whether R has received b_s or $\perp$. This assymmetric lack of knowledge is exactly what is used to do cryptography. Normally, ϵ is one half.

Here is a quick generalization over strings, or over $\mathbb{Z}_n$, using factorization.



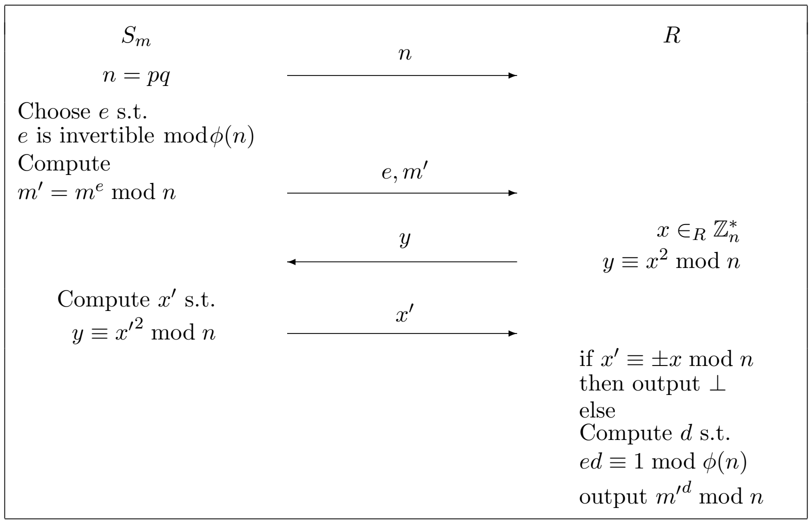
Protocol 6.2: R-OT channel using factorization, abstract

Here m_s is the pair of prime integers (p, q) and with probability one half m_r is equal to (p, q) and with probability one half m_r is an erasure and R knows in which case he his. Here is how to implement this :



Protocol 6.3: R-OT channel using factorization

By mixing this Protocol 6.3 with RSA, we can build a string R-OT.

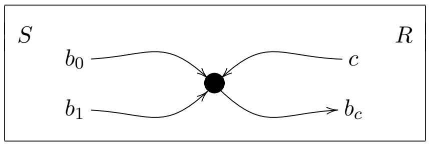


Protocol 6.4: R-OT channel for strings using factorization

Notice however that both of these protocols have the weakness that the Receiver may eventually choose y in such a way that any square root of y may help him factor n. No efficient calculation of such a y is currently known, but we certainly cannot exclude the possibility. Consider for instance, the situation where the Receiver always chooses $y = 1$ and the Sender always computes x' as a non-trivial square root of y; then R receives the message 100% of the time !!! To avoid this loophole, a Zero-Knowledge proof of knowledge of x such that $y = x^2$ can be incorporated together with y, and the Sender can randomize his reply by computing all the square roots of y modolu n and return one of them at random.

6.2 One-out-of-two Oblivious Transfer

We now present the most used and practical oblivious transfer, that is the one out of two oblivious transfer, or $\binom{2}{1}$-OT. This primitive is akin to using a third party T to which the sender gives two bits b_0 and b_1 and to which the receiver gives one bit c. After receiving all three bits, the third party gives the bit b_c to the receiver. In such a protocol, the receiver obviously learns nothing more about $b_{\overline{c}}$ than he might have known before receiving b_c and the senders learns nothing more about the bit c. Hence at the end, the receiver knows perfectly one bit and not the other whilst the sender knows only its input and is clueless about the receivers input. We shall denote $\binom{2}{1}$-OT with the following figure:



Protocol 6.5: One out of two Oblivious Transfer

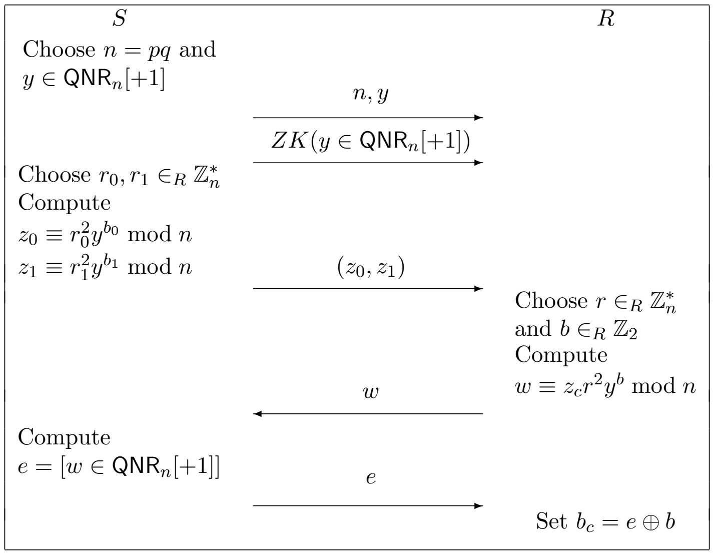
And oblivious transfer has the following requirements :

Completeness:	if S and R are honest, R receives b_c and only b_c.
Obliviousness:	S learns nothing on the choice c
Secrecy:	Only one function of the inputs (b_0, b_1, c) is known to the receiver that is, $f(b_0, b_1, c) = b_c$. For all other functions, the entropy on the answer is either maximal or the output does not depend on $b_{\overline{c}}$

(6.1)

A very powerful theorem tells us that Secrecy only needs to be stated for the xor of bits b_0 and b_1. That is if the entropy on $b_0 \oplus b_1$ is as high as possible, then the entropy of all other functions of b_0 and b_1 is also as high as possible.

Let us see an example based on quadratic residuosity.


S
R
Choose $n = pq$ and $y \in \mathsf{QNR}_n[+1]$
n, y
$ZK(y \in \mathsf{QNR}_n[+1])$
Choose $r_0, r_1 \in_R \mathbb{Z}_n^*$
Compute
$z_0 \equiv r_0^2 y^{b_0} \bmod n$
$z_1 \equiv r_1^2 y^{b_1} \bmod n$
(z_0, z_1)
Choose $r \in_R \mathbb{Z}_n^*$ and $b \in_R \mathbb{Z}_2$
Compute
$w \equiv z_c r^2 y^b \bmod n$
w
Compute
$e = [w \in \mathsf{QNR}_n[+1]]$
e
Set $b_c = e \oplus b$

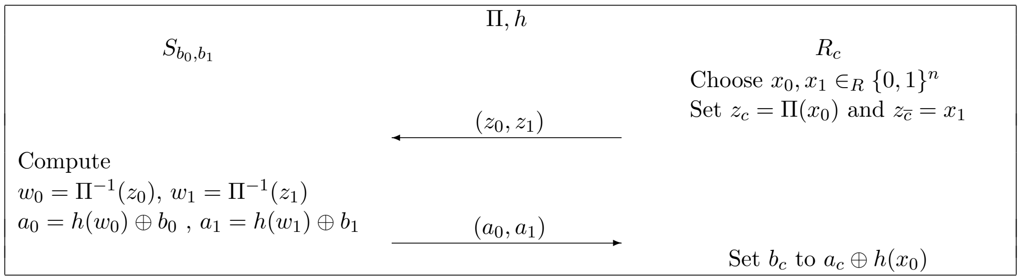
Protocol 6.6: $\binom{2}{1}$-OT based on QNR_n

But how can S be sure that w is well constructed, after all R could try $w \equiv z_0 z_1 r^2 y^b \bmod n$ and thus learning the xor of the bit b_0 and b_1. Hence, R has to do a proof of knowledge proving that w is well constructed and that he knows r and b.

6.2.1 $\binom{2}{1}$-OT from any Random-Self-Reducible Public-key Encryption Scheme

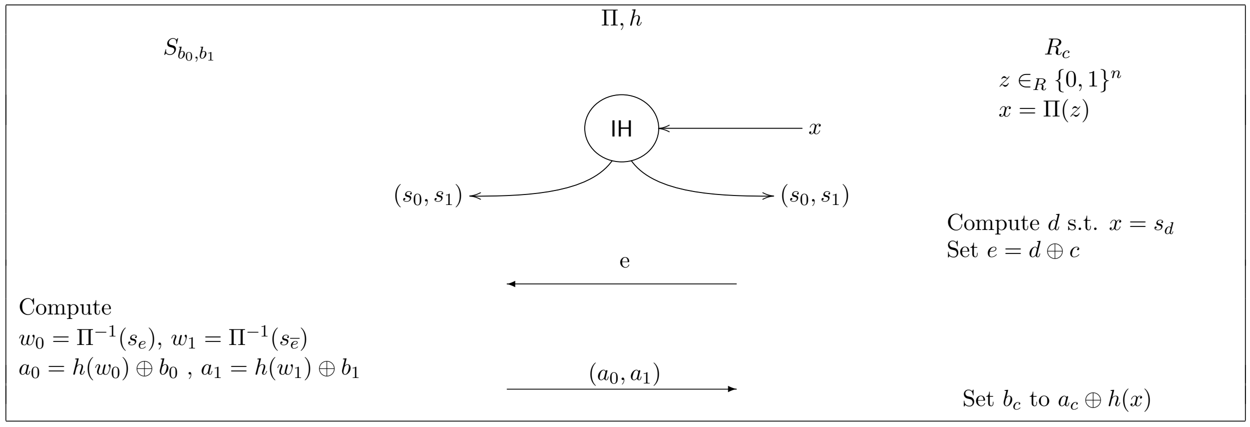
6.2.2 $\binom{2}{1}$-OT from any Trap-door one-way permutation

Protocol 6.6 is a valid protocol that uses very specific properties of quadratic residues in order to implement $\binom{2}{1}$-OT. But as we did in Section 5.4 for bit-commitment, we can build $\binom{2}{1}$-OT from any one-way permutation. Here is a first attempt.


Π, h
S_{b_0,b_1}
R_c
Choose $x_0, x_1 \in_R \{0,1\}^n$
Set $z_c = \Pi(x_0)$ and $z_{\overline{c}} = x_1$
(z_0, z_1)
Compute
$w_0 = \Pi^{-1}(z_0)$, $w_1 = \Pi^{-1}(z_1)$
$a_0 = h(w_0) \oplus b_0$, $a_1 = h(w_1) \oplus b_1$
(a_0, a_1)
Set b_c to $a_c \oplus h(x_0)$

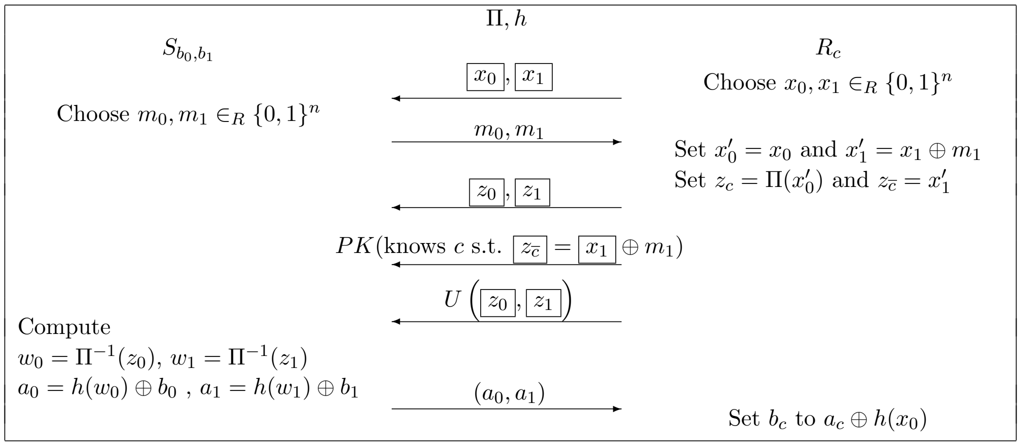
Protocol 6.7: $\binom{2}{1}$-OT attempt based on One-Way Permutation

There are two problems with this protocol. If the receiver is honest, then everything is fine, but if he is actively cheating, then there is no reason for the Receiver not to chose $z_{\overline{c}}$ as $\Pi(x_1)$, and then to retrieve both values. This we can fixed using Interactive Hashing or with some knowledge-proving tricks.


Π, h
S_{b_0,b_1}
R_c
$z \in_R \{0,1\}^n$
$x = \Pi(z)$
IH
x
(s_0, s_1)
(s_0, s_1)
Compute d s.t. $x = s_d$
Set $e = d \oplus c$
e
Compute
$w_0 = \Pi^{-1}(s_e)$, $w_1 = \Pi^{-1}(s_{\overline{e}})$
$a_0 = h(w_0) \oplus b_0$, $a_1 = h(w_1) \oplus b_1$
(a_0, a_1)
Set b_c to $a_c \oplus h(x)$

Protocol 6.8: OT using One-way permutation

The second problem is that the Receiver has to be able to pick elements in the co-domain of Π (or elements in the domain) without knowing their inverse. This sounds strange but think of quadratic residues and the squaring. The squaring function is a permutation over quadratic residues. But how does one choose a quadratic residue without knowing one of its square root? In fact no one knows. Thus, for this protocol to work, one has to require that the permutation be an enhanced one-way permutation. Let us first look at a secure version of the protocol and then give a formal definition of an enhanced one-way permutation.


Π, h
S_{b_0,b_1}
R_c
$\boxed{x_0}, \boxed{x_1}$
Choose $x_0, x_1 \in_R \{0,1\}^n$
Choose $m_0, m_1 \in_R \{0,1\}^n$
m_0, m_1
Set $x'_0 = x_0$ and $x'_1 = x_1 \oplus m_1$
Set $z_c = \Pi(x'_0)$ and $z_{\overline{c}} = x'_1$
$\boxed{z_0}, \boxed{z_1}$
PK(knows c s.t. $\boxed{z_{\overline{c}}} = \boxed{x_1} \oplus m_1$)
$U\left(\boxed{z_0}, \boxed{z_1}\right)$
Compute
$w_0 = \Pi^{-1}(z_0)$, $w_1 = \Pi^{-1}(z_1)$
$a_0 = h(w_0) \oplus b_0$, $a_1 = h(w_1) \oplus b_1$
(a_0, a_1)
Set b_c to $a_c \oplus h(x_0)$

Protocol 6.9: Secure $\binom{2}{1}$-OT based on enhanced One-Way Permutation

Definition 14 (Enhanced One-Way Permutations) *A permutation $f : X \subseteq \{0,1\}^n \longrightarrow X \subseteq \{0,1\}^n$ is said to be enhanced one-way if the three following conditions hold.*

1. *There exists an probabilistic-polynomial time algorithm A such that $f(x) = A(x)$ for all x in the Domain of the function f,*

2. *For all probabilistic-polynomial time algorithm A' we have*

$$\Pr_x[A'(f(x), 1^n) \in f^{-1}(f(x))] \leqslant \mu(n), \tag{6.2}$$

   where x is chosen at random uniformly in the Domain of f and μ is a negligible function.

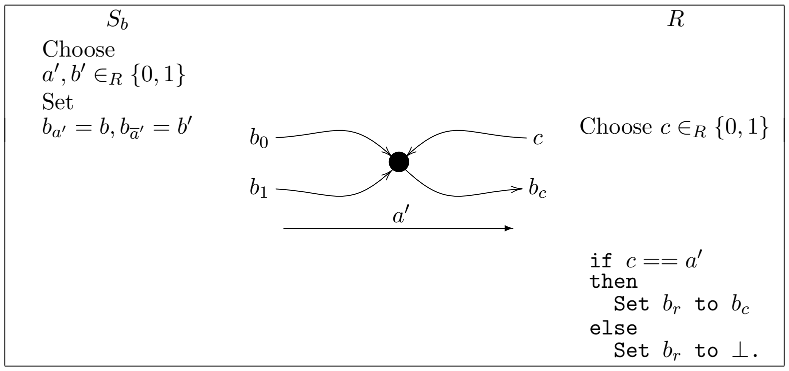
3. *There exists an efficient algorightm I_f that can return an element of the domain of f uniformly distributed over the domain of f.*

Example with Quadratic Residues : Let $n = p * q$ be a product of two large primes p and q such that $p \equiv q \equiv 3 \bmod 4$. For such n's, the function $x^2 \bmod n$ is a permutation of Quadratic Residues into themselves. Unfortunately, this permutation is not "Enhanced" because we do not know an efficient algorithm to select a Quadratic Residue modulo n without knowing its square root...

6.3 Reductions and Applications

6.3.1 Equivalence of Rabin OT and $\binom{2}{1}$-OT

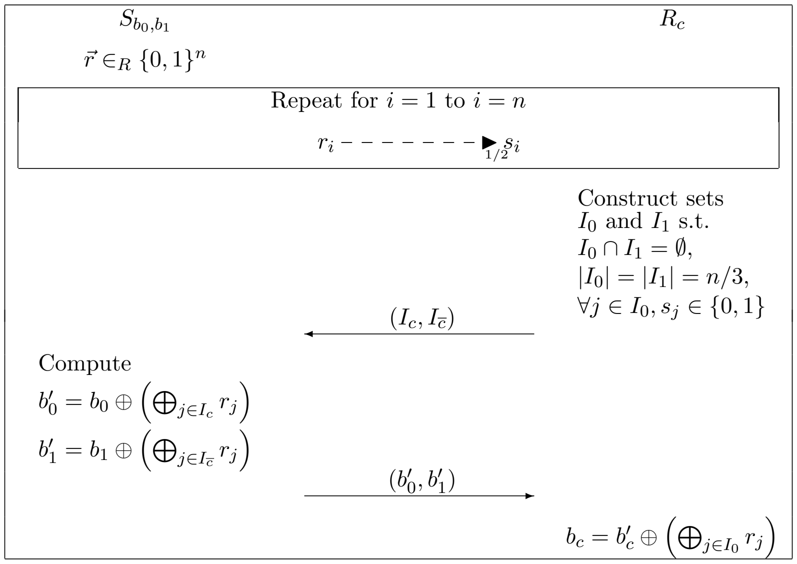
First, we show that One-out-of-two oblivious transfer implies Rabin's oblivious transfer. This is very simple.



Protocol 6.10: Rabin's-OT from a $\binom{2}{1}$-OT

Security is obvious and relies on the security of the $\binom{2}{1}$-OT used in the protocol and this as long as at least one participant is honest.

Reducing $\binom{2}{1}$-OT to Rabins's OT is more involved, especially in the security proof. Here is the Protocol.

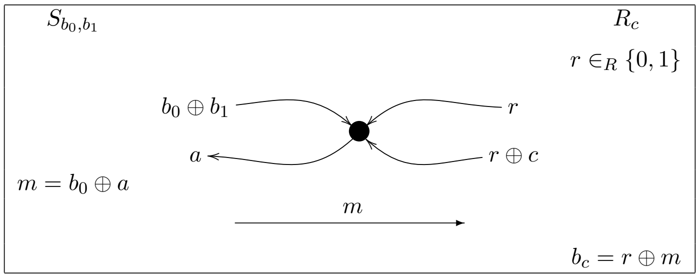

S_{b_0,b_1}
R_c
$\vec{r} \in_R \{0,1\}^n$
Repeat for $i = 1$ to $i = n$
r_i
s_i
$1/2$
Construct sets
I_0 and I_1 s.t.
$I_0 \cap I_1 = \emptyset$,
$|I_0| = |I_1| = n/3$,
$\forall j \in I_0, s_j \in \{0,1\}$
$(I_c, I_{\bar{c}})$
Compute
$b'_0 = b_0 \oplus \left(\bigoplus_{j \in I_c} r_j\right)$
$b'_1 = b_1 \oplus \left(\bigoplus_{j \in I_{\bar{c}}} r_j\right)$
(b'_0, b'_1)
$b_c = b'_c \oplus \left(\bigoplus_{j \in I_0} r_j\right)$

Protocol 6.11: Reducing $\binom{2}{1}$-OT to Rabin's-OT

Discussion of security.

6.3.2 $\binom{2}{1}$-OT is symmetric

If we have two parties A and B and a $\binom{2}{1}$-OT from A to B, then there is a reversing protocol that allows to do $\binom{2}{1}$-OT from B to A (Sometime in the literature, it is denoted TO-$\binom{2}{1}$).



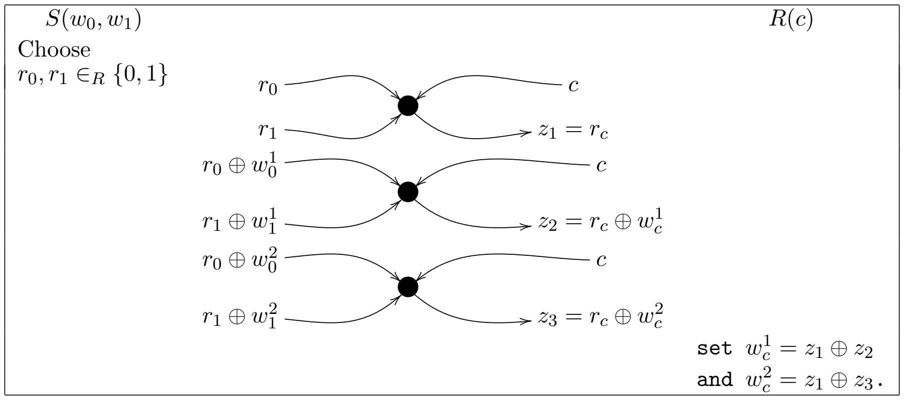
Protocol 6.12: Reducing $\binom{2}{1}$-OT to TO-$\binom{2}{1}$

Let's check that this is indeed a valid $\binom{2}{1}$-OT:

$$r \oplus m = r \oplus b_0 \oplus a = b_0 \oplus r \oplus (r, r \oplus c)_{b_0 \oplus b_1} = b_0 \oplus (0, c)_{b_0 \oplus b_1} = b_0 \oplus ((b_0 \oplus b_1) \wedge c) = b_c$$

6.3.3 $\binom{2}{1}$-OT of strings from $\binom{2}{1}$-OT of bits

Consider the simple example where S wants to send one-out-of-two two-bit string w_0, w_1 to R. Let $z_0 = (r_0, r_0 \oplus w_0^1, w_0^1 \oplus w_0^2)$ and $z_1 = (r_1, r_1 \oplus w_1^1, w_1^1 \oplus w_1^2)$ for random bits r_0, r_1.



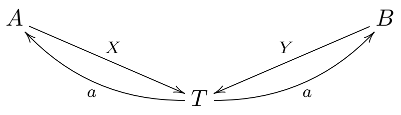
Protocol 6.13: string $\binom{2}{1}$-OT from a bit $\binom{2}{1}$-OT

It is not hard to see that if both parties are honest, R receives w_c and nothing else. It is also not too hard to see (by inspection) that if a dishonest R uses also $\bar{c}$ but only once in one of the three $\binom{2}{1}$-OT, he still gets no information about $w_{\bar{c}}$. Since R cannot use both c and $\bar{c}$ twice out of three runs, he cannot get information on at least one of w_c or $w_{\bar{c}}$.

Indeed, this protocol will work regardless of the size of the values $r_0, w_0^1, w_0^2, r_1, w_1^1, w_1^2$. So to transmit one-out-of-two strings of length 4, we can use protocol 6.13 with string of length 2. This will require 3 executions of one-out-of-two-OT on strings of length 2, each of which will require 3 executions of one-out-of-two-OT of bits. Therefore, using this technique recursively will require 9 one-out-of-two-OT of bits to send strings of length 4. For strings of length 8, 27 one-out-of-two-OT of bits will be used and so on. To send 2^k-bit strings, 3^k $\binom{2}{1}$-OT of bits are sufficient.

6.3.4 A String equality protocol

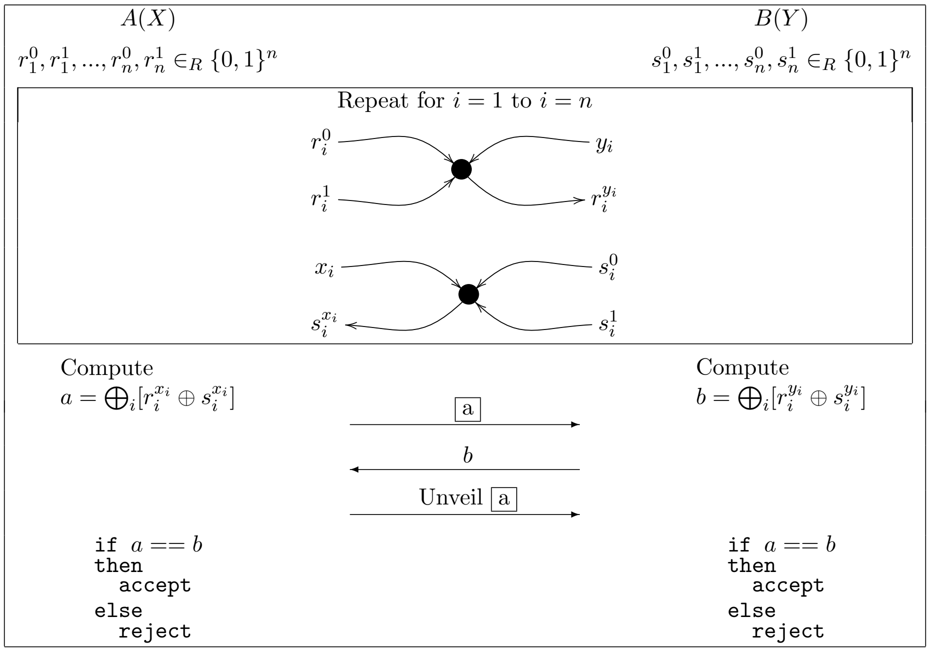
Using a trusted third party, it is very easy to compare two string X and Y belonging to two parties A and B so that they each learn whether X is equal to Y and nothing else.



Protocol 6.14: String equality using a trusted third party

In this protocol, the answer a is one if both strings, X and Y, are equal and zero if they are different. Since trusted third party are a rare commodity, we would be better off having a protocol involving the newly developed primitives. In fact there is a very simple protocol that uses only $\binom{2}{1}$-OT.

In the Following protocol, let the n-bit strings R^j be parsed as $r_1^j r_2^j \ldots r_i^j \ldots r_n^j$. Parse similarly the strings S^0, S^1, X and Y.


$A(X)$
$B(Y)$
$r_1^0, r_1^1, ..., r_n^0, r_n^1 \in_R \{0,1\}^n$
$s_1^0, s_1^1, ..., s_n^0, s_n^1 \in_R \{0,1\}^n$
Repeat for $i = 1$ to $i = n$
r_i^0
r_i^1
y_i
$r_i^{y_i}$
x_i
$s_i^{x_i}$
s_i^0
s_i^1
Compute $a = \bigoplus_i [r_i^{x_i} \oplus s_i^{x_i}]$
Compute $b = \bigoplus_i [r_i^{y_i} \oplus s_i^{y_i}]$
$\boxed{a}$
b
Unveil $\boxed{a}$
if $a == b$ then accept else reject
if $a == b$ then accept else reject

Protocol 6.15: A string equality protocol

In Protocol 6.15 the variables a and b are always equal if X is equal to Y and if A and B are both honest since for every i we have that $r_i^{x_i}$ is equal to $r_i^{y_i}$ and $s_i^{x_i}$ is equal to $s_i^{y_i}$. Let one party be honest, then by simple inspection one can convince himself that if X and Y differ by only one bit, then with probability one half a is not equal to b. Let that bit be bit k, that is x_k is not equal to y_k, but for all other $i \neq k$ we have that $x_i = y_i$. Since, if A is honest, then both R^0 and R^1 are random and for bit k, r_k^0 is not equal to r_k^1, and this with probability one half. What ever B does, the value used to compute a will be different with probability one half from the value used to compute B. Actually for full case analysis using s_i^0 and s_i^1 is necessary, but the idea is the same. Hence, if X and Y are not equal, a and b are different with probability at least one half. One only needs to repeat this protocol to amplify the soundness.

6.4 Two-party computations

6.4.1 Definition of security

We use the following notation: $x \in \mathcal{X}$ denotes the input of the first party, $y \in \mathcal{Y}$ the input of the second party and $z \in \{0,1\}^*$ represents an additional auxiliary input available to both parties but assumed to be ignored by all honest parties. A *g-hybrid protocol* is a pair of (randomized) algorithms $\Pi = (A_1, A_2)$ which can interact by exchanging messages and which additionally have access to the functionality g[1]. More precisely, for a (randomized) function $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{U} \times \mathcal{V}$ the two parties can send x and y to a trusted party and receive u and v, respectively, where $(u, v) = g(x, y)$. Note that a default value is used if a player refuses to send a value. A pair of algorithms $\overline{A} = (\overline{A}_1, \overline{A}_2)$ is called *admissible* for protocol Π if either $\overline{A}_1 = A_1$ or $\overline{A}_2 = A_2$, i.e., if at least one of the parties is honest and uses the algorithm defined by the protocol Π.

[1]Note that g is in general different from f. It should generally be thought of as some trusted cryptographic primitive which the protocol uses as a black box.

Definition 1 (Real Model) *Let $\Pi = (A_1, A_2)$ be a g-hybrid protocol and let $\overline{A} = (\overline{A}_1, \overline{A}_2)$ be an admissible pair of algorithms for the protocol Π. The joint execution of Π under $\overline{A}$ on input pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and auxiliary input $z \in \{0,1\}^*$ in the real model, denoted by*

$$\text{REAL}^g_{\Pi,\overline{A}(z)}(x, y) ,$$

is defined as the output pair resulting from the interaction between $\overline{A}_1(x, z)$ and $\overline{A}_2(y, z)$ using the functionality g.

The *ideal model* defines the optimal scenario where the players have access to an ideal functionality f corresponding to the function they wish to compute. A malicious player may therefore only change (1) his input to the functionality and (2) the output he obtains from the functionality.

Definition 2 (Ideal Model) *The trivial f-hybrid protocol $B = (B_1, B_2)$ is defined as the protocol where both parties send their inputs x and y unchanged to the functionality f and output the values u and v received from f unchanged. Let $\overline{B} = (\overline{B}_1, \overline{B}_2)$ be an admissible pair of algorithms for B. The joint execution of f under $\overline{B}$ in the ideal model on input pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and auxiliary input $z \in \{0,1\}^*$, denoted by*

$$\text{IDEAL}_{f,\overline{B}(z)}(x, y) ,$$

is defined as the output pair resulting from the interaction between $\overline{B}_1(x, z)$ and $\overline{B}_2(y, z)$ using the functionality f.

Any admissible pair of algorithms $\overline{B}$ in the ideal model can be expressed in the following way: the first party receives input (x, z) and the second party receives input (y, z). The two parties produce $(x', z_1) = \overline{B_1^{\text{in}}}(x, z)$ and $(y', z_2) = \overline{B_2^{\text{in}}}(y, z)$, from which x' and y' are inputs to a trusted third party, and z_1 and z_2 are some auxiliary output. The trusted party computes $(u', v') = f(x', y')$ and sends u' to the first party and v' to the second party. The two parties are now given the outputs v' and u' and the auxiliary inputs z_1 and z_2, respectively. The first party outputs $u = \overline{B_1^{\text{out}}}(u', z_1)$ while the second party outputs $v = \overline{B_2^{\text{out}}}(v', z_2)$. Note that if the first party is honest, we have $B_1^{\text{in}}(x, z) = (x, \perp)$ and $B_1^{\text{out}}(u', z_1) = u'$ and similarly for the second party.

Now, to show that a g-hybrid protocol Π securely computes a functionality f, we have to show that anything an adversary can do in the real model can be simulated in the ideal model.

Definition 3 (Perfect Security) *A g-hybrid protocol* Π securely computes f perfectly *if for every pair of algorithms $\overline{A} = (\overline{A}_1, \overline{A}_2)$ that is admissible in the real model for the protocol Π, there exists a pair of algorithms $\overline{B} = (\overline{B}_1, \overline{B}_2)$ that is admissible in the ideal model for protocol B (and where the same players are honest), such that for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $z \in \{0,1\}^*$, we have*

$$\text{IDEAL}_{f,\overline{B}(z)}(x, y) \equiv \text{REAL}^g_{\Pi,\overline{A}(z)}(x, y) .$$

It is sometimes not possible to achieve perfect security. The following definition captures the situation where the simulation has a (small) error ε, defined as the maximal statistical distance between the output distributions in the real and ideal model.

Definition 4 (Statistical Security) *A g-hybrid protocol* Π securely computes f with an error of at most ε *if for every pair of algorithms* $\overline{A} = (\overline{A}_1, \overline{A}_2)$ *that is admissible in the real model for the protocol* Π*, there exists a pair of algorithms* $\overline{B} = (\overline{B}_1, \overline{B}_2)$ *that is admissible in the ideal model for protocol B (and where the same players are honest), such that for all* $x \in \mathcal{X}$*,* $y \in \mathcal{Y}$*, and* $z \in \{0,1\}^*$*, we have*

$$\text{IDEAL}_{f,\overline{B}(z)}(x,y) \equiv_{\varepsilon} \text{REAL}^{g}_{\Pi,\overline{A}(z)}(x,y) \ .$$

The statistical distance is used because it has nice properties and intuitively measures the error of a computation: a protocol Π which securely computes f with an error of at most ε, computes f *perfectly* with probability at least $1 - \varepsilon$.

It is sometimes not possible to achieve statistical security. In those cases we define a computational version of the defition.

Definition 5 (Computational Security) *A g-hybrid protocol* Π securely computes f with a computational error of at most ε *if for every pair of poly-time algorithms* $\overline{A} = (\overline{A}_1, \overline{A}_2)$ *that is admissible in the real model for the protocol* Π*, there exists a pair of poly-time algorithms* $\overline{B} = (\overline{B}_1, \overline{B}_2)$ *that is admissible in the ideal model for protocol B (and where the same players are honest), such that for all* $x \in \mathcal{X}$*,* $y \in \mathcal{Y}$*, and* $z \in \{0,1\}^*$*, we have*

$$\text{IDEAL}_{f,\overline{B}(z)}(x,y) \equiv_{c} \text{REAL}^{g}_{\Pi,\overline{A}(z)}(x,y) \ .$$

Proof example

Consider protocol 6.6 for one-out-of-two OT based on the hardness of Quadratic Residuosity. Let $\overline{A}$ be an adversary to the receiver and consequently let $R = \overline{A}_2$ be a dishonest receiver. We describe the program of a $\overline{B}_2$, the corresponding (trivial) adversary in the IDEAL scenario. As every such trivial adversary, it is broken down in two parts $\overline{B_2^{\text{in}}}$ and $\overline{B_2^{\text{out}}}$.

```
1. Copy fresh random bits to the receiver's random tape and fill up his
   Auxiliary-Input tape.

2. Pick a modulus $n = pq$ at random and a quadratic non-residue $y \in_R QNR_n$.

3. Run a ZK proof that $y \in \mathsf{QNR}_n[+1]$ with R.

4. Choose two random quadratic residues $z_0, z_1$ and send them to R. Wait until R
   issues a ciphertext $w$.

5. Run a proof of knowledge with R to prove that he knows $r$ and $b$ such that
   $w = z_c r y^b \bmod n$; Using the knowledge extractor obtain $r$ and $b$ from R.

6. Compute $c$ such that $w = z_c r y^b \bmod n$.

7. Output $c$ and add $p, q, n, y, b, w$ to the environment, as well as the state $\sigma$ of R.
```

Simulator 6.16: $\overline{B_2^{\text{in}}}$'s simulaton of $\overline{A}_2$

This first simulation corresponds to simulating an execution of the actual protocol upto the point where the ciphertext w is produced by the receiver. Using the knowledge extractor of the proof of knowledge, the simulator determines the bit c indicating which bit the receiver is trying to obtain from the sender.

```
1. Let $u = b_c$ be the output of the functionality $f$ with the sender S.
2. Regardless of the actual decryption of $w$, decrypt $w$ as bit $e = u \oplus b$.
3. Set the state of R to the stored value $\sigma$ and send $e$ to R.
4. Simulate a ZK proof that there exists $s$ such that $w = s^2 y^e \bmod n$ with R.
5. Output whatever $R$ outputs.
```

Simulator 6.17: $\overline{B_2^{\text{out}}}$'s simulaton of $\overline{A}_2$

The second part of the simulation (above) will complete the simulation of a REAL execution to the very last part where the (cheating) receiver outputs whatever he wants (including the entire simulation from his perspective, if he likes).

The resulting execution in the IDEAL scenario will be (only) computationally indistinguishable from a REAL one. This is because the trivial adversary does not know the actual bit $b_{\overline{c}}$ which means that the value used in the simulated protocol will be wrong half the time. Nevertheless, since this bit is never learned by the receiver, it remains computationally indistinguishable from the correct bit.

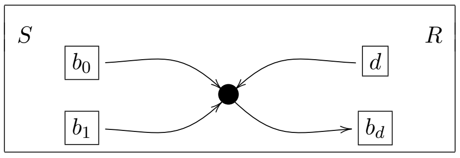
General Composition Theorem

A very important property of the above definitions is that they imply *sequential composition*. The following theorem has been proven in [?].

Theorem 1 *If an h-hybrid protocol Γ securely computes g with an error of at most γ and a g-hybrid protocol Π securely computes f with an error of at most π, then the composed protocol Π^Γ, namely the protocol Π where every call to g is replaced by Γ, is an h-hybrid protocol that securely computes f with an error of at most $\pi + t\gamma$, where t is the number of calls of Π to g.*

6.4.2 General Protocol

We shall now see how to let two parties compute any function f on two strings X and Y where string X is held by party A and string Y is held by party B and this in a secure way. Protocol 6.14 which uses a trusted third party is a good illustration of our goal. That is the answer a of the trusted third party T is the output of the function f on inputs X and Y, or $a = f(X, Y)$. We shall accomplish this using a new primitive which we have not seen yet, *committed oblivious transfer*.

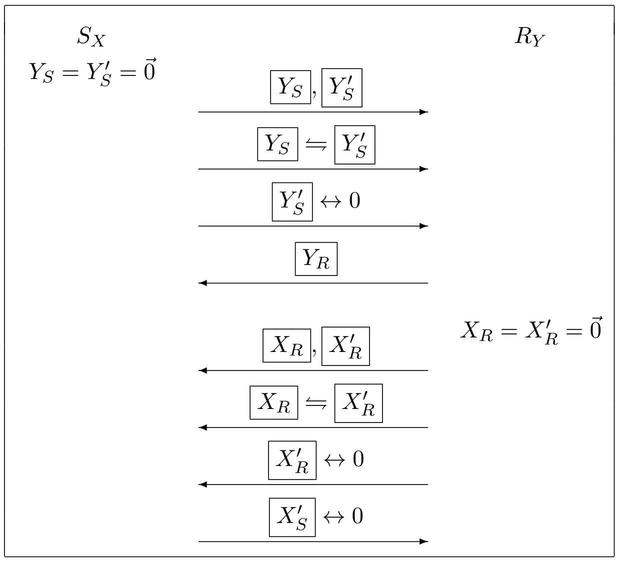


Protocol 6.18: Committed One out of two Oblivious Transfer

The commited oblivious transfer behaves like a normal oblivious transfer but acts on commited input on both sides and ensures that the receiver is committed to the output b_d. Having this device computing any gate on bits x_1 and y_1 is not too complicated. We shall explore in the next section REFREFREF how to build a committed oblivious transfer.

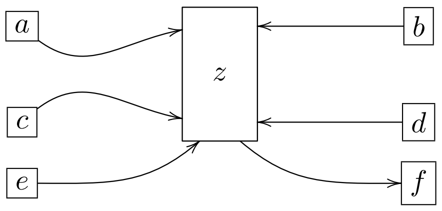
The first thing we shall do is share all informations in the computation between both parties. That is if party S at the onset of the protocol has string X and party R had string Y, we want to share those info in the following way. Party S shall have strings X_S and Y_S and party R strings X_R and Y_R such that $X = X_R \oplus X_S$ amd $Y = Y_R \oplus Y_S$. This initialisation is not hard to do: S can commit to two strings that are all zeros. Prove that they are equal and then open one of them proving that it is uniformly zero. The remaining committed string can be renamed Y_S. And then party R does the same thing to create X_S.

Or, or in a more graphical way:


S_X
R_Y
$Y_S = Y'_S = \vec{0}$
$\boxed{Y_S}, \boxed{Y'_S}$
$\boxed{Y_S} \leftrightharpoons \boxed{Y'_S}$
$\boxed{Y'_S} \leftrightarrow 0$
$\boxed{Y_R}$
$X_R = X'_R = \vec{0}$
$\boxed{X_R}, \boxed{X'_R}$
$\boxed{X_R} \leftrightharpoons \boxed{X'_R}$
$\boxed{X'_R} \leftrightarrow 0$
$\boxed{X'_S} \leftrightarrow 0$

Protocol 6.19: Generalize two party initialization

From then on, things go a little bit as two party computation in chapter 5 section §?? based on the technique of Section §4.6. The two parties have to agree on a circuit that computes the function f they wish to compute gate by gate. They then use the following technique to compute a given gate z on bits x and y, where x and y are one-bit strings and x is S's bits and y is R's bit. Assume that gate z applied to x and y is denothed $z(x, y)$ and assume that x being devided in two part $x = a \oplus b$ where a belongs to S and b belongs to R and $y = c \oplus d$ where c is S's half and d is R's half. And we are looking for $e \oplus f = z((a \oplus b), (c \oplus d))$ where e will be S's half of the answer and f will be R's half of the answer (This is illustrated in Figure). So what we need to compute is the following:

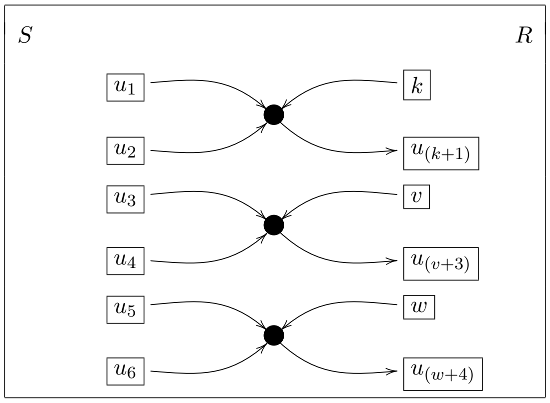


Protocol 6.20: The gate z to be computed

Without loss of generality, let S control the output of the z, by that we mean that S is the party that controls the computation of the gate. We can rewrite the output of the z gate from R's perspective as

$$f = e \oplus z((a \oplus b), (c \oplus d)) \tag{6.3}$$

For S, the only unkown are b and d as a and c are known, and e can just be chosen at random. Hence, for all values of c and d, (there are four possibilities), S can pre-compute the associated value f_{abcde} and let R chose the one it wants. This is were the COT comes in. First, S will generate two random bits u_1 and u_2 and computes the values $u_3 = f_{00} \oplus u_1$, $u_4 = f_{01} \oplus u_1$,$u_5 = f_{10} \oplus u_2$ and $u_6 = f_{11} \oplus u_2$ where f_{ij} is $f_{ij} = e \oplus z((a \oplus i), (c \oplus j))$. Then, S and R engage in three COT where the first one transfer one bit of key, and the two next COT transfer the choice of S acording to the values b and d and the key u_k.



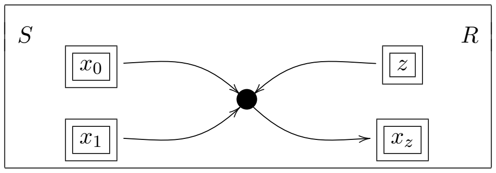
Protocol 6.21: Committed One out of two Oblivious Transfer

Here the values of v and w depends on the value of b and d. if b is zero, the receiver sets v to the value of d in order to pickup the good function output. If b is one, then, w is set to the value of d. Or, in other words, the first COT lets the receiver picks a key that lets him recover one value on only one of the two final COT. If b is zero, then the receiver sets k to zero and retrieves a key that will let him retrieve the value f_{00} or f_{01} depending on the value of d. Obviously, if k is equal to $b = 0$, then the receiver cannot extract anything out of the third COT, as both input bits are encrypted using u_2. Samething goes for $b = 1$.

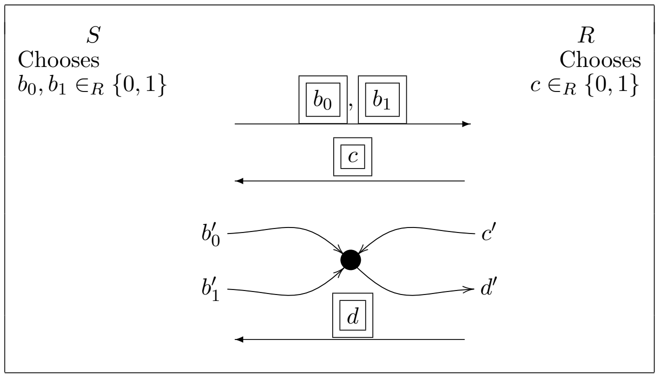
The last thing to add is that the sender has to prove that the commitments u_3, u_4, u_5 and u_6 are all well formed. This can be accomplished with the same function computing technique done before. The sender simply commit to fix vales of b and d and then runs a circuit computing the value of f on $\boxed{abcde}$. So if b and d are zero, then at the end, the receiver is convinced that the value u_2 is the correct one (of course for that the commitment to b and d have to be opened by the sender).

6.5 COT

Now that we know how to use COT, the time has come to learn how to implement such a primitive. We shall assume that we have access to good implementation of the bit-commitment scheme equipped with equality (Perfect equality to simplify things). Let us assume that the sender and the receiver wish to realize the following COT



Protocol 6.22: Goal transaction



Protocol 6.23: Base OT

If S is honest then we can garantee that $b_0 = b'_0$ and $b_1 = b'_1$. If R is honest then we can garantee that $c = c'$ and $d = d'$. We will have to analyze the impact of a dishonest S that uses $b_0 \neq b'_0$ or $b_1 \neq b'_1$ and the impact of a dishonest R that uses $c \neq c'$ or $d \neq d'$. Suppose that S and R participate to a Base OT with random $\boxed{\boxed{b_0}}, \boxed{\boxed{b_1}}, \boxed{\boxed{c}}$ and result $\boxed{\boxed{d}}$. Consider the situation if the committed values are unveiled after the protocol.

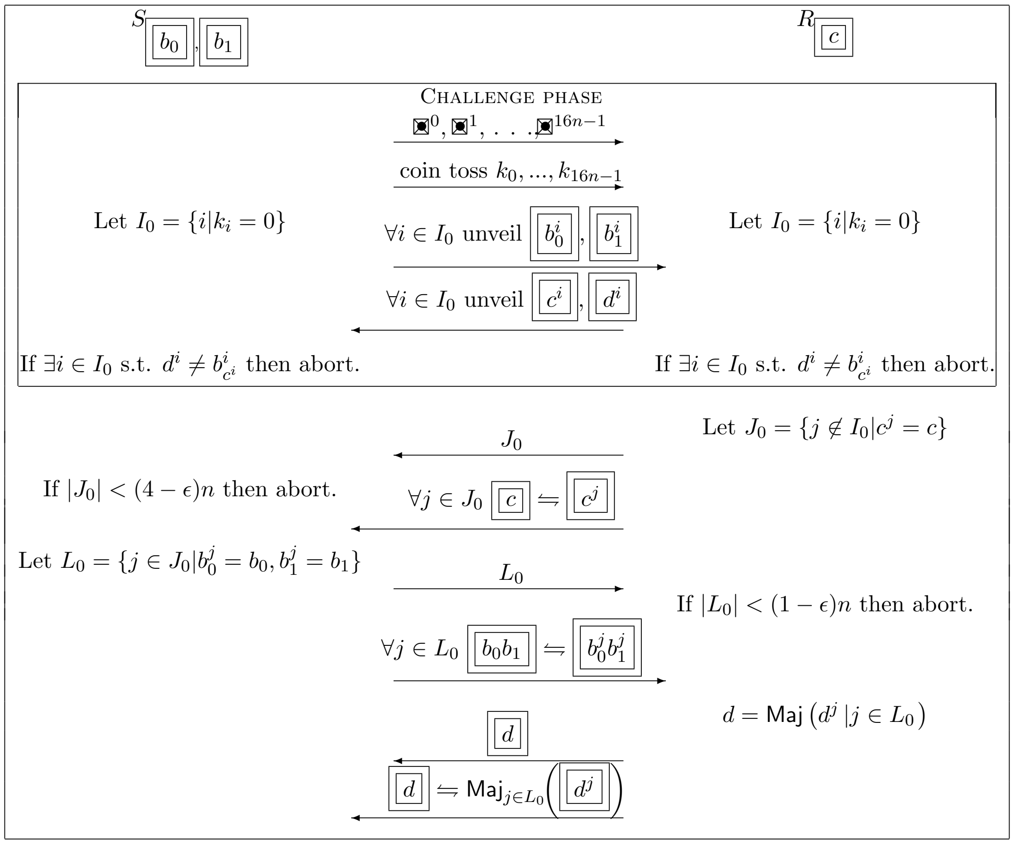
We first analyze the result of a dishonest behavior of S. Suppose $b_0 \neq b'_0$ or $b_1 \neq b'_1$ or both. Honest R uses $c = c'$, $d = d'$ and therefore $d = b'_c$. As a consequence, $d \neq b_c$ exactly when $b_c \neq b'_c$ which occurres with probability at least one half (over random choices of c). If both b_0, b_1 are wrong, then with probability one $d \neq b_c$. Therefore, if S cheats, with probability at least one half, at unveiling of $\boxed{\boxed{b_0}}, \boxed{\boxed{b_1}}, \boxed{\boxed{c}}$ and result $\boxed{\boxed{d}}$, R will notice a discrepency and abort interaction with S.

We now analyze the result of a dishonest behavior of R. Suppose $c \neq c'$. Honest S uses $b_0 = b'_0$ and $b_1 = b'_1$. We can garantee that $d' = b_{c'}$, which means that R has no idea what b_c is. When R commits to d he does it in full ignorance of b_c. After this point, whether he sets $d = d'$ or not makes no difference: whatever d R commits, it has probability one half (over the random choices of b_c) of not being b_c. The case where $c' = c$ and $d \neq d'$ is even more obvious since in that case $d \neq d' = b_c$ will be wrong with certainty at unveiling ! Therefore, if R cheats, with probability at least one half, at unveiling of $\boxed{\boxed{b_0}}, \boxed{\boxed{b_1}}, \boxed{\boxed{c}}$ and result $\boxed{\boxed{d}}$, S will notice a discrepency and abort interaction with R.

We conclude that if a Base OT is executed with random inputs, and if one partie is dishonest, unveiling of the commitments will reveal a discrepency and the honest party will abort with probability at least one half. On the contrary, if a Base OT is executed with random inputs, and if both parties are honest, unveiling of the commitments will reveal no discrepency and no party will abort.

We shall denote Protocol 6.23 by ▣. To resolve the possibility that the committed values are not those used in the internal OT, the parties will execute a large number of Base OT on random committed inputs, parse and process this mass of OT such that at the end, roughly n of them can be used to actually compute a final COT.

Let $\boxtimes^i$ be an OT over values $\boxed{\boxed{b_0^i}}$ $\boxed{\boxed{b_1^i}}$ and choice bit $\boxed{\boxed{c^i}}$ which results in the output $\boxed{\boxed{d^i}}$, were d^i is supposed to be equal to $b^i_{c^i}$.



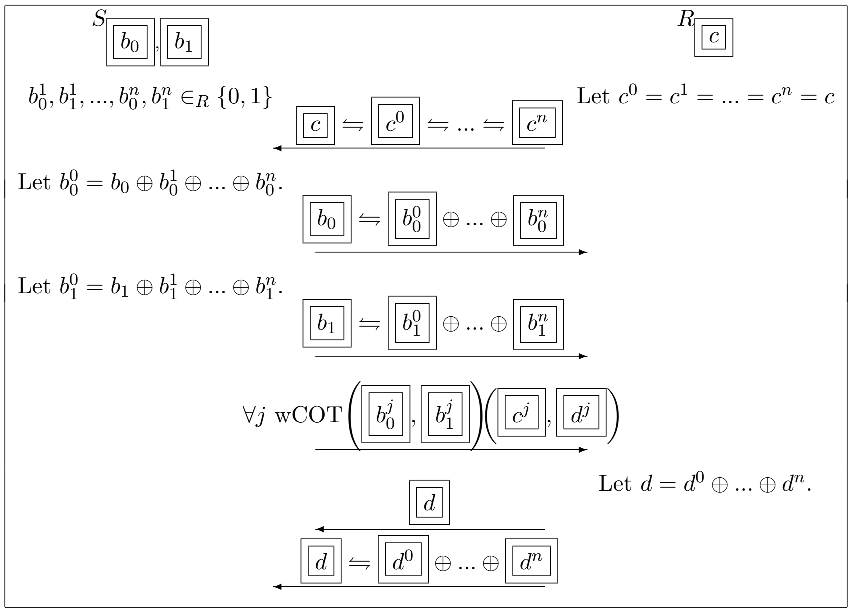
S $\boxed{\boxed{b_0}}, \boxed{\boxed{b_1}}$
R $\boxed{\boxed{c}}$
CHALLENGE PHASE
$\boxtimes^0, \boxtimes^1, \ldots, \boxtimes^{16n-1}$
coin toss $k_0, \ldots, k_{16n-1}$
Let $I_0 = \{i | k_i = 0\}$
Let $I_0 = \{i | k_i = 0\}$
$\forall i \in I_0$ unveil $\boxed{\boxed{b_0^i}}, \boxed{\boxed{b_1^i}}$
$\forall i \in I_0$ unveil $\boxed{\boxed{c^i}}, \boxed{\boxed{d^i}}$
If $\exists i \in I_0$ s.t. $d^i \neq b^i_{c^i}$ then abort.
If $\exists i \in I_0$ s.t. $d^i \neq b^i_{c^i}$ then abort.
Let $J_0 = \{j \notin I_0 | c^j = c\}$
J_0
If $|J_0| < (4-\epsilon)n$ then abort.
$\forall j \in J_0$ $\boxed{\boxed{c}} \leftrightharpoons \boxed{\boxed{c^j}}$
Let $L_0 = \{j \in J_0 | b_0^j = b_0, b_1^j = b_1\}$
L_0
If $|L_0| < (1-\epsilon)n$ then abort.
$\forall j \in L_0$ $\boxed{\boxed{b_0 b_1}} \leftrightharpoons \boxed{\boxed{b_0^j b_1^j}}$
$d = \mathsf{Maj}\left(d^j | j \in L_0\right)$
$\boxed{\boxed{d}}$
$\boxed{\boxed{d}} \leftrightharpoons \mathsf{Maj}_{j \in L_0}\left(\boxed{\boxed{d^j}}\right)$

Protocol 6.24: Reducing weak-COT to BC and $\binom{2}{1}$-OT

The first Reduction 6.24 we define, called weak-COT, builds on BC and $\binom{2}{1}$-OT to nearly acheive COT but with a weakness in the Sender's security: a cheating Receiver may be able to obtain both bits from the Sender at the cost a constant positive probability of causing detection in the CHALLENGE PHASE.

We now describe Protocol 6.24 in details. In a first step, for $0 \leqslant i < 16n$, the sender and receiver shall do $16n$ copies of protocol 6.23. They then choose at random permutation of over the domain of j and permute rename the protocol invocation to $\boxtimes^{0\pi(j)}$. They then open every bit commitment for the first half (that is for $\pi(j) \leqslant n$). The senders opens $\boxed{b_0^{0\pi(j)}}$ and $\boxed{b_1^{0\pi(j)}}$ and the receivers open $\boxed{c^{0\pi(j)}}$ and $\boxed{b_c^{0\pi(j)}}$. And they verify that their data is consistent. That is if, for example, $b_0^{0i} \neq b_1^{0i} = 1$, and $c^{0i} = 1$ then b_c^{0i} should be equal to 1.

So now they have n instances of protocol **??** in which they have some confidence. That was step one. To instill better confidence, they will now regroup these n instances into equivalence classes. As the bit-commitments used are equipped with equality, they can regroup those instances.


S
R
$b_0^1, b_1^1, ..., b_0^n, b_1^n \in_R \{0,1\}$
Let $c^0 = c^1 = ... = c^n = c$
Let $b_0^0 = b_0 \oplus b_0^1 \oplus ... \oplus b_0^n$.
Let $b_1^0 = b_1 \oplus b_1^1 \oplus ... \oplus b_1^n$.
$\forall j$ wCOT
Let $d = d^0 \oplus ... \oplus d^n$.

Protocol 6.25: Reducing COT to wCOT

Problems

6.1 As we have taken the bad habit of providing incomplete protocol, Protocol 6.6 is insecure. Show how the protocol is insecure (give an explicit example of what can be done) and fix it.

6.2 Provide the proof of knowledge needed for Protocol 6.9

6.3 Provide a proof of knowledge for Protocol 6.6 with wich the receiver can convinced the sender that w is well constructed and that the receivers knows c, r and b such that $w \equiv z_c r^2 y^b \bmod n$. Provide the extractor and prove that Protocol 6.6 still obeys the obliviousness property.

6.4 Provide the missing proof of knowledge or Protocol 6.9.

6.5 Prove that Protocol 6.15 is secure. We have done it for one bit of difference between X and Y.

6.6 Prove that if the Rabin's-OT primitive used in Protocol 6.11 is perfect, then Protocol 6.11 is secure.

6.7 Prove that Protocol 6.12 is a valid reduction (you will need to use the $\wedge$ operator).

Chapter 7

Multi-Provers Interactive Proofs

The notion of Multi-Prover Interactive Proofs was introduced by BenOr, Goldwasser, Kilian and Wigderson [?]. In the Two-Prover scenario, we have two provers, Peggy and Patty, that are allowed to share arbitrary information before the proof, but they become physically separated from each other during the execution of the proof, in order to prevent them from communicating. It was demonstrated by Babai, Fortnow, and Lund [?] that Two-Prover Interactive Proofs (with a polynomial-time verifier) exist for all languages in NEXP-time. A fully parallel amalog was achieved by Lapidot and Shamir [?].

A quantum mechanical version of this scenario was considered by Kobayashi, Matsumoto, Yamakami and Yao [?, ?, ?]. To this day, it is still an open problem to establish the exact power of Multi-Prover Quantum Interactive Proofs. A rather vast litterature now exists on this topic (see [?], [?], [?], [?], [?], [?], [?]). However, it is still not even clear whether two provers are as powerful as more-than-two provers.

The Two-Prover Zero-Knowledge Interactive Proofs of [?] rely on the construction of a Bit Commitment scheme, information theoretically secure under the assumption that the provers cannot communicate. We refer the reader to their paper to understand the application of this Bit Commitment scheme to the construction of Two-Prover Zero-Knowledge Proofs. We solely focus on their Bit Commitment scheme for the rest of our work. In this paper, we consider several important questions regarding Two-Prover Bit Commitment schemes. We do not limit our interest of Two-Prover Bit Commitment to the context of Zero-Knowledge proofs; as already discussed in [?] similar techniques lead them to a secure Oblivious Transfer under the same assumption. Given that *any* two-party computation may be achieved from Oblivious Transfer [?], we consider the security of such Bit Commitment scheme in a very general context. We discuss at length the security in a very general composability situation.

In order to argue the security of their Bit Commitment scheme, the authors of [?] asserted the following assumption:

```
"there is no communication between the two provers while interacting with the verifier".
```

The current paper is concerned with the sufficiency of this assertion. We show is Section **??** that, although this assumption *must be made*, it is however considerably too weak, because we exhibit variations of the scheme that are equally binding classically but that are not at all binding if the provers were allowed to share entanglement. It is however a very well known fact that entanglement does not allow communication. Although it is true that they can cheat if they can communicate, it is also true that they can cheat without communicating. Therefore the assumption that the provers cannot communicate is too weak.

This observation can be turned into a purely classical argument by exhibiting a black-box two-party computation, that does not allow them to communicate, but that allows them to cheat the binding condition of the Bit Commitment scheme. This peculiar source of randomness may replace the entanglement used by the attack. Furthermore, the above assertion of can be interpreted as a prescription to the verifier that he should make sure not to help the provers to communicate while interacting with him. Again, this prescription would not prevent him from acting like the black-box we exhibit. Thus, a stronger prescription is mandatory in order to assert security.

We carefully define a notion of *isolation* by which the two provers may not communicate nor perform any non-local sampling beyond what is possible via quantum mechanics. We finally formalize a set of conditions that any third party involved in a Two-Prover Bit Commitment scheme may satisfy to make sure he does not break the assumption that the provers are in isolation. In particular, we make sure that if such a Bit Commitment scheme is used in another larger cryptographic protocol, its security properties will carry over to the larger context.

7.0.1 Related work

The starting point of this research is clearly the Bit Commitment scheme introduced by BenOr, Goldwasser, Kilian and Wigderson [**?**]. The security of a Two-Prover Bit Commitment scheme against quantum adversaries has been considered in the past in the work of Brassard, Crépeau, Mayers and Salvail [**?**]. They showed that if such a Bit Commitment scheme is used in combination to the Quantum Oblivious Transfer protocol of [**?**] it is not sufficient to guarantee the security of the resulting QOT if the two provers can get back together at the end of the protocol. In the current work, we consider only the situation while the provers are isolated.

The research by Cleve, Høyer, Toner and Watrous [**?**] is the main inspiration of the current paper. They have established some relations between Two-Prover Interactive Proofs and so called "non-locality games". More precisely, they showed that certain languages have a classical Two-Prover Interactive Proof that looses soundness if the provers are allowed to share entanglement. Some of our results are very similar to this. However, our new contributions are numerous. While [**?**] focuses on languages, we focus on the tool known as Bit Commitment. This tool is used in many contexts other than proofs of membership to a language: proofs of knowledge, Oblivious Transfer, Zero-Knowledge proofs, general two-party computations. Moreover inspired by the observations of [**?**], we analyze the security of such Two-Prover tools in a completely classical situation. We conclude that proving security of such protocols is very subtle when used in combination with other such tools. We also argue that the claim of security of the protocols of [**?**] requires a lot more assumptions than the mere "no communication" assumption (even in the purely classical situation).

Despite the impossibility theorems of Mayers [**?**] and Lo & Chau [**?**], the possibility of information theoretically secure Bit Commitment schemes in the Two-Prover model is *not* excluded in the classical and quantum models. Indeed, the computations sufficient to cheat the binding condition of a Quantum Bit Commitment scheme in the above "no-go" theorems cannot, in general, be performed by the two provers when they are isolated from each other. This is the reason why these theorems do not apply.

In a closely related piece of work, Kent [**?**] showed how impossibility of communication, implemented through relativistic assumptions, may be used to obtain a Bit Commitment scheme similar to

BGKW that can be constantly updated to avoid cheating. Kent proves the classical security of his scheme while remaining elusive about its quantum security. However, he claims security of one round (see [**?**], Lemma 3, p. 329) of his protocol which is more or less the same as our Lemma **??**. Unfortunately, his proof is incomplete as pointed out in our proof of the Lemma. But we clearly recongnized that he was first to address this question.

A very different set of results [**?**] relates non-locality boxes and two-party protocols such as Bit Commitment and Oblivious Transfer. These are only marginally connected to the current research. They showed how these cryptographic protocols may be securily implemented from those non-locality boxes. On the cotrary, we show how to break such protocols using non-locality boxes...

7.1 Preliminaries

7.1.1 Isolation

First let us define the condition imposed on the two provers: we use the word *isolation* to describe the relation between Peggy and Patty during the protocol. The intuitive meaning of this term is that Peggy and Patty cannot communicate with each other, since this condition is explicitly imposed by the Two-Prover model. However, we introduce this new terminology instead of the traditional "cannot communicate with one another" because we noticed that the meaning of "no-communication" is too weak and must be very clearly defined to produce valid security proofs. This *isolation* will be formally defined in Section **??**. For now, the reader may follow his intuition and picture Peggy and Patty as restricted to compute their messages using only local variables.

7.1.2 Bit Commitment

The primitive known as "Bit Commitment" is a protocol in which a player Alice first sends some information to another player Bob, such that this information *binds* her to a particular bit value b. However, the information sent by Alice is not enough for Bob to learn b (b is *concealed*). At a later time, Alice sends the rest of the information to unveil the bit b, and she cannot change her mind to reveal $\bar{b}$ and convince Bob that this was the value to which she was committed in the first step. The following definitions will be used to characterize the security of a Bit Commitment scheme. Note that the function $\mu(n)$ always refers to a negligible function in n.

Definition 6 *A Bit Commitment scheme is* statistically concealing *if only a negligible amount of information on the committed bit can leak to the verifier before the unveiling stage.*

Definition 7 *A Bit Commitment scheme is statistically binding if, for $b \in \{0,1\}$, the probability p_b that Alice successfully unveils for b satisfies*

$$p_0 + p_1 \leq 1 + \mu(n). \tag{7.1}$$

This binding condition was first proposed by Dumais, Mayers, and Salvail [**?**], as a weaker substitute to the traditional definition $p_b \leq \mu(n)$ for either $b = 0$ or 1. This definition has been henceforward used to show security of many Bit Commitment schemes against quantum adversaries in various models, e.g. [**?**, **?**, **?**].

More recent definitions have been introduced since then ([**?**]) that appear to be better characterization of Bit Commitment security in a quantum setting. However, we have not been able, so far, to find protocols that satisfy these definitions. This, we hope, will be part of future work in this area.