Chapter 4

Knowledge

In this chapter, we investigate what it means for a conversation to "convey" knowledge.

4.1 When Does a Message Convey Knowledge

Our investigation is based on a behavioristic notion of knowledge which models knowledge as the ability to complete a task. A conversation therefore conveys knowledge when the conversation allows the recipient to complete a "new" task that the recipient could not complete before. To quantify the knowledge inherent in a message m, it is therefore sufficient to quantify how much easier it becomes to compute some new function given m.

To illustrate the idea, consider the simplest case of a conversation in when Alice sends a single message to Bob. As before, to describe such phenomena, we must consider a sequence of conversations of increasing size parameterized by n.

Imagine Alice always sends the same message 0^n to Bob. Alice's message is deterministic and it has a short description; Bob can easily produce the message 0^n himself. Thus, this message does not convey any knowledge to Bob.

Now suppose that f is a one-way function, and consider the case when Alice sends Bob the message consisting of "the preimage of the preimage ... (n times) of 0." Once again, the string that Alice sends is deterministic and has a short description.

However, in this case, it is not clear that Bob can produce the message himself because producing the message might require a lot of computation (or a very large circuit). This leads us to a first approximate notion of knowledge. The amount of knowledge conveyed in a message can be quantified by considering the running time and size of a Turing machine that generates the message. With this notion, we can say that any message which can be generated by a constant-sized Turing machine that runs in polynomial-time in n conveys no knowledge since Bob can generate that message himself. These choices can be further refined, but are reasonable for our current purposes.

So far the messages that Alice sends are deterministic; our theory of knowledge should also handle the case when Alice uses randomness to select her message. In this case, the message that Alice sends is drawn from a probability distribution. To quantify the amount of knowledge conveyed by such a message, we again consider the complexity of a Turing machine that can produce the same *distribution* of messages as Alice. In fact, instead of requiring the machine to produce the identical distribution, we may be content with a machine that samples messages from a computationally indistinguishable distribution. This leads to the following informal notion:

"Alice conveys zero knowledge to Bob if Bob can sample from a distribution of messages that is computationally indistinguishable from the distribution of messages that Alice would send."

Shannon's theory of information is certainly closely related to our current discussion; briefly, the difference between information and knowledge in this context is the latter's focus on the *computational* aspects, i.e. running time and circuit size. Thus, messages that convey zero information may actually convey knowledge.

4.2 A Knowledge-Based Notion of Secure Encryption

As a first case study of our behavioristic notion of knowledge, we can re-cast the theory of secure encryption in terms of knowledge. (In fact, this was historically the first approach taken by Goldwasser and Micali.) A good notion for encryption is to argue that an encrypted message conveys zero knowledge to an eavesdropper. In other words, we say that an encryption scheme is secure if the cipertext does not allow the eavesdropper to compute any new (efficiently computable) function about the plaintext message with respect to what she could have computed without the ciphertext message.

The following definition of zero-knowledge encryption[1] captures this very intuition. This definition requires that there exists a simulator algorithm S which produces a string that is indistinguishable from a ciphertext of any message m.

▷**Definition 111.1** (Zero-Knowledge Encryption). A private-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *zero-knowledge encryption scheme* if there exists a p.p.t. simulator algorithm S such that $\forall$ non uniform p.p.t. D, $\exists$ a negligible function $\epsilon(n)$, such that $\forall m \in \{0,1\}^n$ it holds that D distinguishes the following distributions with probability at most $\epsilon(n)$

- $\{k \leftarrow \mathsf{Gen}(1^n) : \mathsf{Enc}_k(m)\}$
- $\{S(1^n)\}$

Note that we can strengthen the definition to require that the above distributions are *identical*; we call the resulting notion *perfect zero-knowledge*.

A similar definition can be used for public-key encryption; here we instead that D cannot distinguish the following two distributions

- $\{pk, sk \leftarrow \mathsf{Gen}(1^n) : pk, \mathsf{Enc}_{pk}(m)\}$
- $\{pk, sk \leftarrow \mathsf{Gen}(1^n) : pk, S(pk, 1^n)\}$

As we show below, for all "interesting" encryption schemes the notion of zero-knowledge encryption is equivalent to the indistinguishability-based notion of secure encryption. We show this for the case of private-key encryption, but it should be appreciated that the same equivalence (with essentially the same proof)

[1]This is a variant of the well-known notion of semanical security.

holds also for the case of public-key encryption. (Additionally, the same proof show that perfect zero-knowledge encryption is equivalent to the notion of perfect secrecy.)

▷**Theorem 112.2** *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme such that* $\mathsf{Gen}, \mathsf{Enc}$ *are both p.p.t, and there exists a polynomial-time machine* M *such that for every* n*,* $M(n)$ *outputs a messages in* $\{0,1\}^n$*. Then* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is secure if and only if it is zero-knowledge.*

Proof. We prove each direction separately.

Security implies ZK. Intuitively, if it were possible to extract "knowledge" from the encrypted message, then there would be a way to distinguish between encryptions of two different messages. More formally, suppose that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure. Consider the following simulator $S(1^n)$:

1. Pick a message $m \in \{0,1\}^n$ (recall that by our asumptions, this can be done in p.p.t.)
2. Pick $k \leftarrow \mathsf{Gen}(1^n)$, $c \leftarrow \mathsf{Enc}_k(m)$.
3. Output c.

It only remains to show that the output of S is indistinguishable from the encryption of any message. Assume for contradiction that there exist a n.u. p.p.t. distinguisher D and a polynomial $p(\cdot)$ such that for infinitely many n, there exist some message m'_n such that D distinguishes

- $\{k \leftarrow \mathsf{Gen}(1^n) : \mathsf{Enc}_k(m_n)\}$
- $\{S(1^n)\}$

with probability $p(n)$. Since $\{S(1^n)\} = \{k \leftarrow \mathsf{Gen}(1^n); m'_n \leftarrow M(1^n) : \mathsf{Enc}_k(m'_n)\}$, it follows that there exists messages m_n and m'_n such that their encryptions can be distinguished with inverse polynomial probability; this contradict the security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

ZK implies Security. Suppose for the sake of reaching contradiction that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is zero-knowledge, but there exists a n.u. p.p.t. distringuisher D and a polynomial $p(n)$, such that for infinitely many n there exist messages m_n^1 and m_n^2 such that D distinguishes

- $H_n^1 = \{k \leftarrow \mathsf{Gen}(1^n) : \mathsf{Enc}_k(m_n^1)\}$
- $H_n^2 = \{k \leftarrow \mathsf{Gen}(1^n) : \mathsf{Enc}_k(m_n^2)\}$

with probability $p(n)$. Let S denote the zero-knowledge simulator for $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and define the hybrid distribution H_3:

- $H_n^3 = \{S(1^n)\}$

By the hybrid lemma, D distinguishes between either H_n^1 and H_n^2 or between H_n^2 and H_n^3, with probability $\frac{1}{2p(n)}$ for infinitely many n; this is a contradiction. □

4.3 Zero-Knowledge Interactions

So far, we have only worried about an honest Alice who wants to talk to an honest Bob, in the presence of a malicious Eve. We will now consider a situation in which neither Alice nor Bob trust each other.

Suppose Alice (the prover) would like to convince Bob (the verifier) that a particular string x is in a language L. Since Alice does not trust Bob, Alice wants to perform this proof in such a way that Bob learns nothing else except that $x \in L$. In particular, it should not be possible for Bob to later prove that $x \in L$ to someone else. For instance, it might be useful in a cryptographic protocol for Alice to show Bob that a number N is the product of exactly two primes, but without revealing anything about the two primes.

It seems almost paradoxical to prove a theorem in such a way that the theorem proven cannot be established subsequently. However, *zero-knowledge proofs* can be used to achieve exactly this property.

Consider the following toy example involving the popular "Where's Waldo?" children's books. Each page is a large complicated illustration, and somewhere in it there is a small picture of

Waldo, in his sweater and hat; the reader is invited to find him. Sometimes, you wonder if he is there at all.

The following protocol allows a prover to convince a verifier that Waldo is in the image without revealing any information about where he is in the image: Take a large sheet of newsprint, cut a Waldo-sized hole, and overlap it on the "Where's waldo" image, so that Waldo shows through the hole. This shows he is somewhere in the image, but there is no extra contextual information to show where.

A slightly more involved example follows. Suppose you want to prove that two pictures or other objects are distinct without revealing anything about the distinction. Have the verifier give the prover one of the two, selected at random. If the two really are distinct, then the prover can reliably say "this one is object 1", or "this is object 2". If they were identical, this would be impossible.

The key insight in both examples is that the verifier generates a puzzle related to the original theorem and asks the prover to solve it. Since the puzzle was generated by the verifier, the verifier already knows the answer—the only thing that the verifier does learn is that the puzzle *can* be solved by the prover, and therefore the theorem is true.

4.4 Interactive Protocols

To begin the study of zero-knowledge proofs, we must first formalize the notion of interaction. The first step is to consider an *Interactive Turing Machine*. Briefly, an interactive Turing machine (ITM) is a Turing machine with a read-only *input* tape, a read-only *auxiliary input* tape, a read-only *random* tape, a read/write *work-tape*, a read-only communication tape (for receiving messages) a write-only communication tape (for sending messages) and finally an *output* tape. The content of the input (respectively auxiliary input) tape of an ITM A is called *the input* (respectively *auxiliary input*) *of* A and the content of the output tape of A, upon halting, is called *the output of* A.

A protocol (A, B) is a pair of ITMs that share communication tapes so that the (write-only) send-tape of the first ITM is the (read-only) receive-tape of the second, and vice versa. The computation of such a pair consists of a sequence of rounds $1, 2, \ldots$. In each round only one ITM is active, and the other is idle. A round ends with the active machine either halting —in which case the protocol ends— or by it entering a special *idle* state. The string m written on the communication tape in a round is called the *message sent* by the active machine to the idle machine.

In this chapter, we consider protocols (A, B) where both A and B receive the *same* string as input (but not necessarily as auxiliary input); this input string is the *common input* of A and B. We make use of the following notation for protocol executions.

Executions, transcripts and views. Let M_A and M_B be vectors of strings $M_A = \{m_A^1, m_A^2, \ldots\}$, $M_B = \{m_B^1, m_B^2, \ldots\}$ and let $x, r_1, r_2, z_1, z_2 \in \{0,1\}^*$. We say that the pair

$$((x, z_1, r_1, M_A), (x, z_2, r_2, M_B))$$

is an execution of the protocol (A, B) if, running ITM A on common input x, auxiliary input z_1 and random tape r_1 with ITM B on x, z_2 and r_2, results in m_A^i being the i^{th} message received by A and in m_B^i being the i^{th} message received by B. We also denote such an execution by $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$.

In an execution $((x, z_1, r_1, M_A), (x, z_2, r_2, M_B)) = (V_A, V_B)$ of the protocol (A, B), we call V_A the *view of* A (in the execution), and V_B the *view of* B. We let $\mathsf{view}_A[A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)]$ denote A's view in the execution $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$ and $\mathsf{view}_B[A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)]$ B's view in the same execution. (We occasionally find it convenient referring to an execution of a protocol (A, B) as a *joint view* of (A, B).)

In an execution $((x, z_1, r_1, M_A), (x, z_2, r_2, M_B))$, the tuple (M_A, M_B) is called the transcript of the execution.

Outputs of executions and views. If e is an execution of a protocol (A, B) we denote by $\mathsf{out}_X(e)$ the output of X, where $X \in \{A, B\}$. Analogously, if v is the view of A, we denote by $\mathsf{out}(v)$ the output of A in v.

Random executions. We denote by $A(x, z_1) \leftrightarrow B(x, z_2)$, the probability distribution of the random variable obtained by selecting each bit of r_1 (respectively, each bit of r_2, and each bit of r_1 and r_2) randomly and independently, and then outputting $A_{r_1}(x, z_1) \leftrightarrow B_{r_2}(x, z_2)$. The corresponding probability distributions for view and out are analogously defined.

Time Complexity of ITMs. We say that an ITM A has time-complexity $t(n)$, if for every ITM B, every common input x, every auxiliary inputs z_a, z_b, it holds that $A(x, z_a)$ *always* halts within $t(|x|)$ steps in an interaction with $B(x, z_b)$, regardless of the content of A and B's random tapes). Note that time complexity is defined as an upperbound on the running time of A *independently* of the content of the messages it receives. In other words, the time complexity of A is the *worst-case* running time of A in *any* interaction.

4.5 Interactive Proofs

With this notation, we start by considering *interactive proofs* in which a prover wishes to convince a verifier that a statement is true (without consideration of the additional property of zero-knowledge). Roughly speaking, we require the following two properties from an interactive proof system: it should be possible for a prover to convince a verifier of a true statment, but it should not be possible for a malicious prover to convince a verifier of a false statement.

▷**Definition 116.1** (Interactive Proof). A pair of interactive machines (P, V) is an interactive proof system for a language L if V is a p.p.t. machine and the follwing properties hold.

1. (Completeness) For every $x \in L$, there exists a witness string $y \in \{0,1\}^*$ such that for every auxiliary string z:

$$\Pr\left[\mathsf{out}_V[P(x, y) \leftrightarrow V(x, z)] = 1\right] = 1$$

2. (Soundness) There exists some negligible function ϵ such that for all $x \notin L$ and for all prover algorithms P^*, and all

auxiliary strings $z \in \{0,1\}^*$,

$$\Pr\left[\mathsf{out}_V[P^*(x) \leftrightarrow V(x,z)] = 0\right] > 1 - \epsilon(|x|)$$

Note that the prover in the definition of an interactive proof need not be efficient. (Looking forward, we shall later consider a definition which requires the prover to be efficient.)

Note that we do not provide any auxilary input to the "malicious" prover strategy P^*; this is without loss of generality as we consider *any* prover strategy; in particular, this prover strategy could have the auxilary input hard-coded.

Note that we can relax the definition and replace the $1 - \epsilon(|x|)$ with some constant (e.g., $\frac{1}{2}$); more generally we say that an interactive proof has soundness error $s(n)$ if it satisfies the above definitiob, but with $1 - \epsilon(|x|)$ replaced by $1 - s(n)$.

The class of languages having an interactive proofs is denoted **IP**. It trivially holds that **NP** $\subset$ **IP**—the prover can simply provide the NP witness to the verifier, and the verifier checks if it is a valid witness. Perhaps surprisingly, there are languages that are not known to be in NP that also have interactive proofs: as shown by Shamir, every language in **PSPACE**—i.e., the set of languages that can be recognized in polynomial space—has an interactive proof; in fact, **IP** = **PSPACE**. Below we provide an example of an interactive proof for a language that is not known to be in **NP**. More precisely, we show an interactive proof for the Graph Non-isomorphism Language.

An Interactive Proof Graph Non-isomorphism

A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E which consists of pairs of verticies. Typically, we use n to denote the number of verticies in a graph, and m to denote the number of edges. Recall that two graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ are isomorphic if there exists a permutation σ over the verticies of V_1 such that $V_2 = \{\sigma(v_1) \mid v_1 \in V_1\}$ and $E_2 = \{(\sigma(v_1), \sigma(v_2)) \mid (v_1, v_2) \in E_1\}$. In other words, permuting the verticies of G_1 and maintaining the permuted edge relations results in the graph G_2. We will often write $\sigma(G_1) = G_2$ to indicate that graphs G_1 and G_2 are isomorphic via the permutation σ. Similarly, two graphs are non-isomorphic if there exists no permutation σ for which $\sigma(G_1) = G_2$. (See Fig. 2 for examples.)

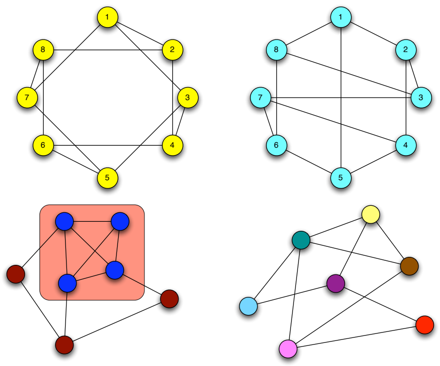

Figure 118.2: (a) Two graphs that are isomorphic, (b) Two graphs that are non-isomorphic. Notice that the highlighted 4-clique has no corresponding 4-clique in the extreme right graph.

Notice that the language of isomorphic graphs is in **NP** since the permutation serves as a witness. Let L_{niso} be the language of pairs of graphs (G_0, G_1) that have the same number of verticies but are not isomorphic. This language $L_{\text{niso}} \in$ **coNP** and is not known to be in **NP**. Consider the following protocol 118.3 which proves that two graphs are non-isomorphic.

PROTOCOL 118.3: PROTOCOL FOR GRAPH NON-ISOMORPHISM

Input:	$x = (G_0, G_1)$ where $\|G_i\| = n$
$V \xrightarrow{H} P$	The verifier, $V(x)$, chooses a random bit $b \in \{0,1\}$, chooses a random permutation $\sigma \in S_n$, computes $H \leftarrow \sigma(G_b)$, and finally sends H to the prover.
$V \xleftarrow{b'} P$	The prover computes a b' such that H and $G_{b'}$ are isomorphic and sends b' to the verifier.
$V(x, H, b, b')$	The verifier accepts and outputs 1 if $b' = b$ and 0 otherwise.
	Repeat the procedure $\|G_1\|$ times.

▷**Proposition 118.4** *Protocol 118.3 is an interactive proof for L_{niso}.*

Proof. Completeness follows by inspection: If G_1 and G_2 are not isomorphic, then the Prover (who runs in exponential time in this protocol) will always succeed in finding b' such that $b' = b$. For soundness, the prover's chance of succeeding in one iteration of the basic protocol is $1/2$. This is because when G_1 and G_2 are isomorphic, then H is independent of the bit b. Since each iteration is independent of all prior iterations, the probability that a cheating prover succeeds is therefore upper-bounded by 2^{-n}. □

4.5.1 Interactive proofs with Efficient Provers

The Graph Non-isomorphism protocol required an exponential-time prover. Indeed, a polynomial time prover would imply that $L_{\text{niso}} \in \mathbf{NP}$. In cryptographic applications, we require protocols in which the prover is efficient. To do so we are required to restrict our attention to languages in **NP**; the prover strategy should be efficient when given a **NP**-witness y to the statement x that it attempts to prove. See Appendix B for a formal definition of **NP** languages and witness relations.

▷**Definition 119.5** (Interactive Proof with Efficient Provers). An interactive proof system (P, V) is said to have an *efficient prover* with respect to the witness relation R_L if P is p.p.t. and the completeness condition holds for every $y \in R_L(x)$.

Note that although we require that the honest prover strategy P is efficient, the soundness condition still requires that not even an all powerful prover strategy P^* can cheat the verifier V. A more relaxed notion—called an *interactive argument* considers only P^*'s that are n.u. p.p.t.

Although we have already shown that the Graph Isomorphism Language has an Interactive Proof, we now present a new protocol which will be useful to us later. Since we want an efficient prover, we provide the prover the witness for the theorem $x \in L_{\text{iso}}$, i.e., we provide the permutation to the prover.

An Interactive Proof for Graph Isomorphism

PROTOCOL 120.6: PROTOCOL FOR GRAPH ISOMORPHISM

Input:	$x = (G_0, G_1)$ where $\|G_i\| = n$
P's **witness**:	σ such that $\sigma(G_0) = G_1$
$V \xleftarrow{H} P$	The prover chooses a random permutation π, computes $H \leftarrow \pi(G_0)$ and sends H.
$V \xrightarrow{b} P$	The verifier picks a random bit b and sends it.
$V \xleftarrow{\gamma} P$	If $b = 0$, the prover sends π. Otherwise, the prover sends $\gamma = \pi \cdot \sigma^{-1}$.
V	The verifier outputs 1 if and only if $\gamma(G_b) = H$.
P, V	Repeat the procedure $\|G_1\|$ times.

▷**Proposition 120.7** *Protocol 120.6 is an interactive proof for L_{niso}.*

Proof. If the two graphs G_1, G_2 are isomorphic, then the verifier always accepts because $\pi(H) = G_1$ and $\sigma(\pi(H)) = \sigma(G_1) = G_2$. If the graphs are not isomorphic, then no malicious prover can convince V with probability greater than $\frac{1}{2}$: if G_1 and G_2 are not isomorphic, then H can be isomorphic to at most one of them. Thus, since b is selected at random after H is fixed, then with probability $\frac{1}{2}$ it will be the case that H and G_i are not isomorphic. This protocol can be repeated many times (provided a fresh H is generated), to drive the probability of error as low as desired. □

As we shall see, the Graph-Isomorphism protocol is in fact also zero-knowledge.

4.6 Zero-Knowledge Proofs

In addition to being an interactive proof, the protocol 120.6 also has the property that the verifier "does not learn anything" beyond the fact that G_0 and G_1 are isomorphic. In particular, the verifier does not learn anything about the permutation σ. As discussed in the introduction, by "did not learn anything," we mean that the verifier is not able to perform any extra tasks after seeing a proof that $(G_0, G_1) \in L_{\text{iso}}$. As with zero-knowledge encryption, we can formalize this idea by requiring there to be a simulator algorithm that produces "interactive transcripts" that are identical to the transcripts that the verifier encounters during the actual execution of the interactive proof protocol.

▷**Definition 121.1 (Honest Verifier Zero-Knowledge)** *Let (P, V) be an efficient interactive proof for the language $L \in$ **NP** with witness relation R_L. (P, V) is said to be* honest verifier zero-knowledge *if there exists a p.p.t. simulator S such that for every n.u. p.p.t. distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that for every $x \in L, y \in R_L(x), z \in \{0,1\}^*$, D distinguishes the following distributions with probability at most $\epsilon(n)$.*

- $\{\mathsf{view}_V[P(x,y) \leftrightarrow V(x,z)]\}$
- $\{S(x,z)\}$

Intuitively, the definition says whatever V "saw" in the interactive proof could have been generated by V himself by simply running the algorithm $S(x,z)$. The auxiliary input z to V denotes any a-priori information V has about x; as such the definition requires that V does not learn anything "new" (even considering this a-priori information).

This definition is, however, not entirely satisfactory. It ensures that when the verifier V follows the protocol, it gains no additional information. But what if the verifier is malicious and uses some other machine V^*. We would still like V to gain no additional information. To achieve this we modify the definition to require the existence of a simulator S for every, possibly malicious, efficient verifier strategy V^*. For technical reasons, we additionally slighty weaken the requirement on the simulator S and only require it to be an *expected p.p.t*—namely a machine whose *expected running-time* (where expectation is taken only over the internal randomness of the machine) is polynomial.[2]

▷**Definition 121.2 (Zero-knowledge)** *Let (P, V) be an efficient interactive proof for the language $L \in$ **NP** with witness relation R_L. (P, V) is said to be* zero-knowledge *if for every p.p.t. adversary V^* there exists an expected p.p.t. simulator S such that for every n.u. p.p.t.*

[2]In essence, this relaxation will greatly facilitate the construction of zero-knowledge protocols

distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that for every $x \in L, y \in R_L(x), z \in \{0,1\}^*$, D distinguishes the following distributions with probability at most $\epsilon(n)$.

- $\{\mathsf{view}_{V^*}[P(x,y) \leftrightarrow V^*(x,z)]\}$
- $\{S(x,z)\}$

Note that here only consider p.p.t. adversaries V^* (as opposed to *non-uniform* p.p.t. adversaries). This only makes our definition stronger: V^* can anyway receive any non-uniform "advice" as its auxiliary input; in contrast, we can now require that the simulator S is only p.p.t. but is also given the auxiliary input of V^*. Thus, our definition says that even if V^* is non-uniform, the simulator only needs to get the same non-uniform advice to produce its transcript.

In the case of zero-knowledge encryption, we can strengthen the definition to require the above two distributions to be identically distributed; in this case the interactive proof is called perfect zero-knowledge.

An alternate formalization more directly considers what V^* "can do", instead of what V^* "sees". That is, we require that whatever V^* can do after the interactions, V^* could have already done it before it. This is formalized by simply exchanging view_{V^*} to out_{V^*} in the above definition. We leave it as an exercise to the reader to verify that the definitions are equivalent.

We can now show that the Graph-isomorphism protocol is zero-knowledge.

▷**Theorem 122.3** *Protocol 120.6 is a perfect zero-knowledge interactive proof for the Graph-isomorphism language (for some canonical witness relation).*

Proof. We have already demonstrated completeness and soundness in Proposition 120.7. We show how to construct an expected p.p.t. simulator for every p.p.t. verifier V^*. $S(x,z)$ makes use of V^* and proceeds as described in Algorithm 123.4. For simplicity, we here only provide a simulator for a single iteration of the Graph Isomorphism protocol; the same technique easily extends to the iterated version of the protocol as well. In fact, as we show in §7.2.1, this holds for every zero-knowledge protocol: namely, the sequential repetition of any zero-knowledge protocol is still zero-knowledge.

123.4: Simulator for Graph Isomorphism

```
1. Randomly pick $b' \leftarrow \{0,1\}$, $\pi \leftarrow S_n$
2. Compute $H \leftarrow \pi(G_{b'})$.
3. Emulate the execution of $V^*(x,z)$ by feeding it $H$ and truly
   random bits as its random coins; let $b$ denote the response
   of $V^*$.
4. If $b = b'$ then output the view of $V^*$—i.e., the messages
   $H, \pi$, and the random coins it was feed. Otherwise, restart
   the emulation of $V^*$ and repeat the procedure.
```

We need to show the following properties:

- the expected running time of S is polynomial,
- the output distribution of S is correctly distributed.

Towards this goal, we start with the following lemma.

▷**Lemma 123.5** *In the execution of $S(x,z)$, H is identically distributed to $\pi(G_0)$, and* $\Pr[b' = b] = \frac{1}{2}$.

Proof. Since G_0 is an isomorphic copy of G_1, the distribution of $\pi(G_0)$ and $\pi(G_1)$ is the same for random π. Thus, the distribution of H is independent of b'. In particular, H has the same distribution as $\pi(G_0)$.

Furthermore, since V^* takes only H as input, its output, b, is also independent of b'. As b' is chosen at random from $\{0,1\}$, it follows that $\Pr[b' = b] = \frac{1}{2}$. □

From the lemma, we directly have that S has probability $\frac{1}{2}$ of succeeding in each trial. It follows that the expected number of trials before terminating is 2. Since each round takes polynomial time, S runs in expected polynomial time.

Also from the lemma, H has the same distribuion as $\pi(G_0)$. Thus, if we were always able to output the corresponding π, then the output distribution of S would be the same as in the actual protocol. However, we only output H if $b' = b$. Fortunetly, since H is independent from b', this does not change the output distribution. □

4.7 Zero-knowledge proofs for NP

We now show that every language in **NP** has a zero-knowledge proof system assuming the existence of a one-way permutation. (In fact, using a more complicated proof, it can be shown that general one-way functions suffice.)

▷**Theorem 124.1** *If one-way permutations exist, then every language in* **NP** *has a zero-knowledge proof.*

Proof. Our proof proceeds in two steps:

Step 1: Show a ZK proof (P', V') (with efficient provers) for an **NP**-complete language; the particular language we will consider is *Graph 3 Coloring*—namely the language of all graphs whose vertices can be colored using only three colors $1, 2, 3$ such that no two connected vertices have the same color.

Step 2: To get a zero-knowledge proof (P, V) for any **NP** language, proceed as follows: Given a language L, instance x and witness y, both P and V reduce x into an instance of a Graph 3-coloring x'; this can be done using Cook's reduction (the reduction is deterministic which means that both P and V will reach the same instance x). Additionally, Cook's reduction can be applied to the witness y yielding a witness y' for the instance x'. The parties then execute protocol (P, V) on common input x', and the prover additionally uses y' as its auxiliary input.

It is easy to verify that the above protocol is a zero-knowledge proof if we assume that (P', V') is a zero-knowledge proof for Graph 3-coloring. Thus it remains to show a zero-knowledge proof for Graph 3-coloring.

To give some intuition, we start by proving a "physical" variant of the protocol. Given a graph $G = (V, E)$, where V is the set of verticies, and E is the set of edges, and a coloring C of the vertices V, the prover picks a random permutation π over the colors $\{1,2,3\}$ and physically colors the graph G with the permuted colors. It then covers each vertices with individual cups. The verifier is next asked to pick a random edge, and the prover is supposed to remove the two cups corresponding to the vertices of the edge, to show that the two vertices have different colors. If they don't the prover has been caught cheating, otherwise the interaction is repeated (each time letting the prover pick a new random permutation π.) As we shall see, if the procedure is repeated $O(n|E|)$, where $|E|$ is the number of edges, then the soundness error will be 2^{-n}. Additionally, in each round of the interaction, the verifier only learns something he knew before—two random (but different) colors. See Figure 2 for an illustration of the protocol.

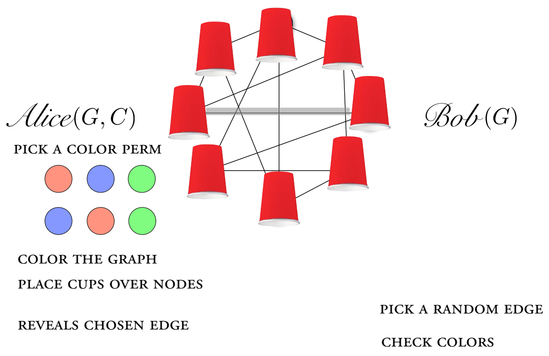

Figure 125.2: 3-Coloring

To be able to digitally implement the above protocol, we need to have a way to implement the "cups". Intuitively, we require two properties from the cups: a) the verifier should not be able to see what is under the cup—i.e., the cups should be *hiding*, b) the prover should not be able to change what is under a cup—i.e, the cups should be *binding*. The cryptographic notion that achieves both of these properties is a *commitment scheme*.

4.7.1 Commitment Schemes

Commitment schemes are usually referred to as the digital equivalent of a "physical" locked box. They consist of two phases:

Commit phase : Sender puts a value v in a locked box.

Reveal phase : Sender unlocks the box and reveals v.

We require that before the reveal phase the value v should remain hidden: this property is called *hiding*. Additionally, during the reveal phase, there should only exists a signle value that the commitment can be revealed to: this is called *binding*.

We provide a formalization of single-messages commitments where both the commit and the reveal phases only consist of a single message sent from the committer to the receiver.

▷**Definition 126.3** (Commitment). A polynomial-time machine Com is called a commitment scheme it there exists some polynomial $\ell(\cdot)$ such that the following two properties hold:

1. (Binding): For all $n \in \mathbb{N}$ and all $v_0, v_1 \in \{0,1\}^n$, $r_0, r_1 \in \{0,1\}^{l(n)}$ it holds that $\mathsf{Com}(v_0, r_0) \neq \mathsf{Com}(v_1, r_1)$.
2. (Hiding): For every n.u. p.p.t. distinguisher D, there exists a negligible function ϵ such that for every $n \in N$, $v_0, v_1 \in \{0,1\}^n$, D distinguishes the following distributions with probability at most $\epsilon(n)$.
   - $\{r \leftarrow \{0,1\}^{l(n)} : \mathsf{Com}(v_0, r)\}$
   - $\{r \leftarrow \{0,1\}^{l(n)} : \mathsf{Com}(v_1, r)\}$

Just as the definition of multi-message secure encryption, we can define a notion of "multi-value security" for commitments. It directly follows by a simple hybrid argument that any commitment scheme is multi-value secure.

▷**Theorem 126.4** *If one-way permutations exist, then commitment schemes exist.* ■

Proof. We construct a single-bit commitment scheme using a one-way permutation. A full-fledge commitment scheme to a value $v \in \{0,1\}^n$ can be obtained by individually committing to each bit of v; the security of the full-fledged construction follows as a simple application of the hybrid lemma (show this!).

Let f be a one-way permutation with a hard-core predicate h. Let $\mathsf{Com}(b, r) = f(r), b \oplus h(r)$. It directly follows from the construction that Com is binding. Hiding follows using identically the same proof as in the proof of Theorem 104.4. □

4.7.2 A Zero-knowledge Proof for Graph 3-Coloring

We can now replace the physical cups with commitments in the protocol for Graph 3-coloring described above. Consider the following protocol.

PROTOCOL 127.5: ZERO-KNOWLEDGE FOR GRAPH 3-COLORING

Common input:	$G = (V, E)$ where $\|V\| = n$, $\|E\| = m$
Prover input:	Witness $y = c_0, c_1, \ldots, c_m$
$P \to V$	Let π be a random permutation over $\{1, 2, 3\}$. For each $i \in [1, n]$, the prover sends a commitment to the color $\pi(c_i) = c'_i$.
$V \to P$	The verifier sends a randomly chosen edge $(i, j) \in E$
$P \to V$	The prover opens commitments c'_i and c'_j
V	V accepts the proof if and only if $c'_i \neq c'_j$
P, V	Repeat the procedure $n\|E\|$ times.

▷**Proposition 127.6** *Protocol 127.5 is a zero-knowledge protocol for the language of 3-colorable graphs.*

Proof. The completeness follows by inspection. If G is not 3 colorable, then for each coloring $c_1, \ldots, c_m$, there exists at least one edge which has the same colors on both endpoints. Thus, soundness follows by the binding property of the commitment scheme: In each iteration, a cheating prover is caught with probability $1/|E|$. Since the protocol is repeated $|E|^2$ times, the probability of successfully cheating in all rounds is

$$\left(1 - \frac{1}{|E|}\right)^{n|E|} \approx e^{-n}$$

For the zero-knowledge property, the prover only "reveals" 2 random colors in each iteration. The hiding property of the commitment scheme intuitively guarantees that "everything else" is hidden.

To prove this formally requires more care. We construct the simulator in a similar fashion to the graph isomorphism simulator. Again, for simplicity, we here only provide a simulator for a single iteration of the Graph 3-Coloring protocol. As previously mentioned, this is without loss of generality (see §7.2.1).

128.7: Simulator for Graph 3-Coloring

1. Pick a random edge $(i', j') \in E$ and pick random colors $c'_i, c'_j \in \{1, 2, 3\}, c'_i \neq c'_j$. Let $c'_k = 1$ for all other $k \in [m] \setminus \{i', j'\}$
2. Just as the honest prover, commit to c'_i for all i and feed the commitments to $V^*(x, z)$ (while also providing it truly random bits as its random coins).
3. Let (i, j) denote the answer from V^*.
4. If $(i, j) = (i', j')$ reveal the two colors, and output the view of V^*. Otherwise, restart the process from the first step, but at most $n|E|$ times.
5. If, after $n|E|$ repetitions the simulation has not been sucessful, output `fail`.

By construction it directly follows that S is a p.p.t. We proceed to show that the simulator's output distribution is correctly distributed.

▷**Proposition 128.8** *For every n.u. p.p.t. distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that for every $x \in L, y \in R_L(x), z \in \{0, 1\}^*$, D distinguishes the following distributions with probability at most $\epsilon(n)$.*

— $\{\mathsf{view}_{V^*}[P(x, y) \leftrightarrow V^*(x, z)]\}$

— $\{S(x, z)\}$

Assume for contradiction that there exists some n.u. distinguisher D and a polynomial $p(\cdot)$, such that for infinitely many $x \in L, y \in R_L(x), z \in \{0, 1\}^*$, D distinguishes

- $\{\mathsf{view}_{V^*}[P(x,y) \leftrightarrow V^*(x,z)]\}$
- $\{S(x,z)\}$

with probability $p(|x|)$. First consider the following hybrid simulator S' that receives the real witness $y = c_1, \ldots, c_n$ (just like the Prover): S' proceeds exactly as S, except that instead of picking the colors c'_i and c'_j at random, it picks π at random and lets $c'_i = \pi(c_i)$ and $c'_j = \pi(c_j)$ (just as the prover). It directly follows that $\{S(x,z)\}$ and $\{S'(x,z,y)\}$ are identically distributed.

Next, consider the following hybrid simulator S'': S'' proceeds just as S, but just as the real prover commits to a random permutation of the coloring given in the witness y; except for that it does everything just like S—i.e., it picks i', j' at random and restarts if $(i,j) \neq (i',j')$. If we assume that S' never outputs `fail`, then clearly the following distributions are identical.

- $\{\mathsf{view}_{V^*}[P(x,y) \leftrightarrow V^*(x,z)]\}$
- $\{S''(x,z,y)\}$

However, as i,j and i',j' are independently chosen, S' fails with probability

$$\left(1 - \frac{1}{|E|}\right)^{n|E|} \approx e^{-n}$$

It follows that

- $\{\mathsf{view}_{V^*}[P(x,y) \leftrightarrow V^*(x,z)]\}$
- $\{S''(x,z,y)\}$

can be distinguished with probability at most $O(e^{-n}) < \frac{1}{2p(n)}$. By the hybrid lemma, D thus distinguishes $\{S'(x,z,y)\}$ and $\{S''(x,z)\}$ with probability $\frac{1}{2p(n)}$.

Next consider a sequence $S_0, S_1, \ldots, S_{2n|E|}$ of hybrid simulators where S_k proceeds just as S' in the first k iterations, and like S'' in the remaining ones. Note that $\{S_0(x,z,y)\}$ is identically distributed to $\{S''(x,z,y)\}$ and $\{S_{2n|E|}(x,z,y)\}$ is identically distributed to $\{S'(x,z,y)\}$. By the hybrid lemma, there exist some k such that D distinguishes between

- $\{S_k(x, z, y)\}$
- $\{S_{k+1}(x, z, y)\}$

with probability $\frac{1}{2n|E|p(n)}$. (Recall that the only difference between S_k and S_{k+1} is that in the $k+1$th iteration, S_k commits to $1'$s, whereas S_{k+1} commits to the real witness.) Consider, next, another sequence of $6|E|$ hybrid simulators $\tilde{S}_0, \ldots, \tilde{S}_{6|E|}$ where $\tilde{S}_e$ proceeds just as S_k if in the $k+1$th iteration the index, of the edge (i', j') and permutation π, is smaller than e; otherwise, it proceeds just as S_{k+1}. Again, note that $\{\tilde{S}_0(x, z, y)\}$ is identically distributed to $\{S_{k+1}(x, z, y)\}$ and $\{\tilde{S}_{6|E|}(x, z, y)\}$ is identically distributed to $\{S_k(x, z, y)\}$. By the hybrid lemma there exists some $e = (\tilde{i}, \tilde{j}, \tilde{\pi})$ such that D distinguishes

- $\{\tilde{S}_e(x, z, y)\}$
- $\{\tilde{S}_{e+1}(x, z, y)\}$

with probability $\frac{1}{12n|E|^2p(n)}$. Note that the only difference between $\{\tilde{S}_e(x, z, y)\}$ and $\{\tilde{S}_{e+1}(x, z, y)\}$ is that in $\{\tilde{S}_{e+1}(x, z, y)\}$, if in the kth iteration $(i, j, \pi) = (\tilde{i}, \tilde{j}, \tilde{\pi})$, then V^* is feed commitments to $\pi(c_k)$ for all $k \notin \{i, j\}$, whereas in $\{\tilde{S}_{e+1}(x, z, y)\}$, it is feed commitments to 1. Since $\tilde{S}_e$ is computable in n.u. p.p.t, by closure under efficient operations, this contradicts the (multi-value) computational hiding property of the commitments. □

4.8 Proof of knowledge

4.9 Applications of Zero-knowledge

One of the most basic applications of zero-knowledge protocols are for secure identification to a server. A typical approach to identification is for a server and a user to share a secret password; the user sends the password to the server to identify herself. This approach has one major drawback: an adversary who intercepts this message can impersonate the user by simply "replaying" the password to another login session.

It would be much better if the user could prove identity in such a way that a *passive adversary* cannot subsequently impersonate the user. A slightly better approach might be to use a signature. Consider the following protocol in which the User and Server share a signature verification key V to which the User knows the secret signing key S.

1. The User sends the Server the "login name."
2. The server sends the User the string σ="Server name, r" where r is a randomly chosen value.
3. The user responds by signing the message σ using the signing key S.

Proving Your Identity without Leaving a Trace

In the above protocol, the "User" is trying to prove to "Server" that she holds the private key S corresponding to a public key V; r is a nonce chosen at random from $\{0,1\}^n$. We are implicitly assuming that the signature scheme resists chosen-plaintext attacks. Constraining the text to be signed in some way (requiring it to start with "server") helps.

This protocol has a subtle consequence. The server can prove that the user with public key V logged in, since the server has, and can keep, the signed message $\sigma = \{$"Server name", $r\}$. This property is sometimes undesirable. Imagine that the user is accessing a politically sensitive website. With physical keys, there is no way to prove that a key has been used. Here we investigate how this property can be implemented with cryptography.

In fact, a zero-knowledge protocol can solve this problem. Imagine that instead of sending a signature of the message, the User simply proves in zero-knowledge that it knows the key S corresponding to V. Certainly such a statement is in an **NP** language, and therefore the prior protocols can work. Moreover, the server now has no reliable way of proving to another party that the user logged in. In particular, no one would believe a server who claimed as such because the server could have easily created the "proof transcript" by itself by running the Simulator algorithm. In this way, zero-knowledge protocols provide a tangibly new property that may not exist with simple "challenge-response" identity protocols.

Chapter 7

Composability

7.1 Composition of Encryption Schemes

7.1.1 CCA-Secure Encryption

So far, we have assumed that the adversary only captures the ciphertext that Alice sends to Bob. In other words, the adversary's attack is a *ciphertext only* attack. One can imagine, however, stronger attack models. We list some of these models below:

Attack models:

- Known plaintext attack – The adversary may get to see pairs of form $(m_0, Enc_k(m_0)) \ldots$
- Chosen plain text (CPA) – The adversary gets access to an encryption oracle before and after selecting messages.
- Chosen ciphertext attack

  CCA1: ("Lunch-time attack") The adversary has access to an encryption oracle and to a decryption oracle before selecting the messages. (due to Naor and. Yung)

  CCA2: This is just like a CCA1 attack except that the adversary also has access to decryption oracle after selecting the messages. It is not allowed to decrypt the challenge ciphertext however. (introduced by Rackoff and Simon)

Fortunately, all of these attacks can be abstracted and captured by a simple definition which we present below. The different attacks can be captured by allowing the adversary to have *oracle*-access to a special function which allows it to mount either CPA, CCA1, or CCA2- attacks.

▷**Definition 168.1 (CPA/CCA-Secure Encryption)** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme. Let the random variable* $\mathsf{IND}_b^{O_1,O_2}(\Pi, \mathcal{A}, n)$ *where* $\mathcal{A}$ *is a non-uniform p.p.t.,* $n \in \mathbb{N}$, $b \in \{0,1\}$ *denote the output of the following experiment:*

```
$\mathsf{IND}_b^{O_1,O_2}(\Pi, ma, n)$
    $k \leftarrow \mathsf{Gen}(1^n)$
    $m_0,\ m_1,\ state \leftarrow A^{O_1(k)}(1^n)$
    $c \leftarrow \mathsf{Enc}_k(m_b)$
    Output $A^{O_2(k)}(c, state)$
```

Then we say π *is CPA/CCA1/CCA2 secure if* $\forall$ *non-uniform p.p.t. A:*

$$\left\{\mathsf{IND}_0^{O_1,O_2}(\pi, A, n)\right\}_n \approx \left\{\mathsf{IND}_1^{O_1,O_2}(\pi, A, n)\right\}_n$$

where O_1 *and* O_2 *are defined as:*

CPA	$[\mathsf{Enc}_k;\ \mathsf{Enc}_k]$
CCA1	$[\mathsf{Enc}_k, \mathsf{Dec}_k;\ \mathsf{Enc}_k]$
CCA2	$[\mathsf{Enc}_k, \mathsf{Dec}_k;\ \mathsf{Enc}_k, \mathsf{Dec}_k]$

Additionally, in the case of CCA2 attacks, the decryption oracle returns $\perp$ *when queried on the challenge ciphertext c.*

7.1.2 A CCA1-Secure Encryption Scheme

We will now show that the encryption scheme presented in construction 99.1 satisfies a stronger property than claimed earlier. In particular, we show that it is CCA1 secure (which implies that it is also CPA-secure).

▷**Theorem 168.2** π *in construction 99.1 is CPA and CCA1 secure.*

Proof. Consider scheme $\pi^{RF} = (\mathsf{Gen}^{RF}, \mathsf{Enc}^{RF}, \mathsf{Dec}^{RF})$, which is derived from π by replacing the PRF f_k in π by a truly random function. π^{RF} is CPA and CCA1 secure. Because the adversary only has access to the encryption oracle after chosing m_0 and m_1, the only chance adversary can differentiate $\mathsf{Enc}_k(m_0) = r_0||m_0 \oplus f(r_0)$ and $\mathsf{Enc}_k(m_1) = r_1||m_1 \oplus f(r_1)$ is that the encryption oracle happens to have sampled the same r_0 or r_1 in some previous query, or additionally, in a CCA1 attack, the attacker happens to have asked decryption oracle to decrypt ciphertext like $r_0||m$ or $r_1||m$. All cases have only negligible probabilities.

Given π^{RF} is CPA and CCA1 secure, then so is π. Otherwise, if there exists one distinguisher D that can differentiate the experiment results ($\mathsf{IND}_0^{\mathsf{Enc}_k;\mathsf{Enc}_k}$ and $\mathsf{IND}_1^{\mathsf{Enc}_k;\mathsf{Enc}_k}$ in case of CPA attack, while $\mathsf{IND}_0^{\mathsf{Enc}_k,\mathsf{Dec}_k;\mathsf{Enc}_k}$ and $\mathsf{IND}_1^{\mathsf{Enc}_k,\mathsf{Dec}_k;\mathsf{Enc}_k}$ in case of CCA1 attack) then we can construct another distinguisher which internally uses D to differentiate PRF from truly random function. □

7.1.3 A CCA2-Secure Encryption Scheme

However, the encryption scheme π is not CCA2 secure. Consider the attack: in experiment $\mathsf{IND}_b^{\mathsf{Enc}_k,\mathsf{Dec}_k;\mathsf{Enc}_k,\mathsf{Dec}_k}$, given ciphertext $r\|c \leftarrow \mathsf{Enc}_k(m_b)$, the attacker can ask the decryption oracle to decrypt $r\|c+1$. As this is not the challenge itself, this is allowed. Actually $r\|c+1$ is the ciphertext for message m_b+1, as

$$\begin{aligned}\mathsf{Enc}_k(m_b+1) &= (r\|(m_b+1)) \oplus f_k(r)\\ &= r\|m_b \oplus f_k(r) + 1\\ &= r\|c+1\end{aligned}$$

Thus the decryption oracle would reply m_b+1. The adversary can differentiate which message's encryption it is given.

We construct a new encryption scheme that is CCA2 secure. Let $\{f_s\}$ and $\{g_s\}$ be families of PRF on space $\{0,1\}^{|s|} \to \{0,1\}^{|s|}$.

ALGORITHM 169.3: π' : MANY-MESSAGE CCA2-SECURE ENCRYPTION

Assume $m \in \{0,1\}^n$ and let $\{f_k\}$ be a PRF family

$\mathsf{Gen}'(1^n)$: $k_1, k_2 \leftarrow U_n$

$\mathsf{Enc}'_{k_1,k_2}(m)$: Sample $r \leftarrow U_n$. Set $c_1 \leftarrow m \oplus f_{k_1}(r)$. Output the ciphertext $(r, c_1, f_{k_2}(c))$

$\mathsf{Dec}'_{k_1,k_2}((r, c_1, c_2))$: If $f_{k_2}(c_1) \neq c_2$, then output $\perp$. Otherwise output $c_1 \oplus f_{k_1}(r)$

▷**Theorem 170.4** *π' is CCA2-secure.*

Proof. The main idea is to prove by contradiction. In specific, if there is an CCA2 attack on π', then there is an CPA attack on π, which would contradict with the fact that π is CPA secure.

A CCA2 attack on π' is a p.p.t. machine A', s.t. it can differentiate $\left\{\mathsf{IND}_0^{\mathsf{Enc}_k,\mathsf{Dec}_k;\mathsf{Enc}_k,\mathsf{Dec}_k}\right\}$ and $\left\{\mathsf{IND}_1^{\mathsf{Enc}_k,\mathsf{Dec}_k;\mathsf{Enc}_k,\mathsf{Dec}_k}\right\}$. Visually, it works as that in figure **??**. The attacker A' needs accesses to the Enc'_k and Dec'_k oracles. To devise a CPA attack on π, we want to construct another machine A as depicted in figure **??**. To leverage the CCA2 attacker A', we simulate A as in figure **??** which internally uses A'.

Formally, the simulator works as follows:

- Whenever A' asks for an encryption of message m, A asks its own encryption oracle Enc_{s_1} to compute $c_1 \leftarrow \mathsf{Enc}_{s_1}(m)$. However A' expects an encryption of the form $c_1||c_2$ which requires the value s_2 to evaluate $g_{s_2}(c_1)$; A does not have access to s_2 and so instead computes $c_2 \leftarrow \{0,1\}^n$, and replies $c_1||c_2$.

- Whenever A' asks for a decryption $c_1||c_2$. If we previously gave A' $c_1||c_2$ to answer an encryption query of some message m, then reply m, otherwise reply $\perp$.

- Whenever A' outputs m_0, m_1, output m_0, m_1.

- Upon receiving c, feed $c||r$, where $r \leftarrow \{0,1\}^n$ to A'.

- Finally, output A''s output.

Consider the encryption scheme $\pi'^{RF} = (\mathsf{Gen}'^{RF}, \mathsf{Enc}'^{RF}, \mathsf{Dec}'^{RF})$ which is derived from π' by replacing every appearance of g_{s_2} with a truly random function.

Note that the simulated Enc' is just Enc'^{RF}, and Dec' is very similar to Dec'^{RF}. Then A' inside the simulator is nearly conducting CCA2 attack on π'^{RF} with the only exception when A' asks an $c_1||c_2$ to Dec' which is not returned by a previous encryption query and is a correct encryption, in which case Dec' falsely returns $\perp$. However, this only happens when $c_2 = f(c_1)$, where f is the truly random function. Without previous encryption query, the attacker can only guess the correct value of $f(c_1)$ w.p. $\frac{1}{2^n}$, which is negligible.

Thus we reach that: if A' breaks CCA2 security of π'^{RF}, then it can break CPA security of π. The premise is true as by assumption A' breaks CCA2 security of π', and that PRF is indistinguishable from a truly random function. □

7.1.4 CCA-secure Public-Key Encryption

We can also extend the notion of CCA security to public-key encryption schemes. Note that, as the adversary already knows the the public key, there is no need to provide it with an encryption oracle.

▷**Definition 171.5 (CPA/CCA-Secure Public Key Encryption)** *If the triplet $\Pi = (\mathsf{Gen}, \mathsf{Enc}, Dec)$ is a public key encryption scheme, let the random variable $\mathsf{Ind}_b(\Pi, \mathcal{A}, 1^n)$ where $\mathcal{A}$ is a non-uniform p.p.t. adversary, $n \in \mathbb{N}$, and $b \in \{0,1\}$ denote the output of the following experiment:*

```
$\mathsf{Ind}_b(\Pi, \mathcal{A}, n)$
    $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$
    $m_0,\ m_1,\ state \leftarrow A^{O_1(sk)}(1^n, pk)$
    $c \leftarrow \mathsf{Enc}_{pk}(m_b)$
    $Output\ A^{O_2(k)}(c, state)$
```

We say that Π is CPA/CCA1/CCA2 secure if for all non-uniform p.p.t. $\mathcal{A}$, the following two distributions are computationally indistinguishable:

$$\{\mathsf{Ind}_0(\Pi, \mathcal{A}, n)\}_{n\in\mathbb{N}} \approx \{\mathsf{Ind}_1(\Pi, \mathcal{A}, n)\}_{n\in\mathbb{N}}$$

The oracles O_1, O_2 are defined as follows:

CPA	$[\cdot,\cdot]$
CCA1	$[\mathsf{Dec},\cdot]$
CCA2	$[\mathsf{Dec},\mathsf{Dec}^*]$

where Dec^* *answers all queries except for the challenge ciphertext c.*

It is not hard to see that the encryption scheme in Construction 104.3 is CPA secure. CCA2 secure public-key encryption schemes are, however, significantly hard to construct; such contructions are outside the scope of this chapter.

7.1.5 Non-Malleable Encryption

Until this point we have discussed encryptions that prevent a passive attacker from discovering any information about messages that are sent. In some situations, however, we may want to prevent an attacker from creating a new message from a given encryption.

Consider an auction for example. Suppose the Bidder Bob is trying to send a message containing his bid to the Auctioneer Alice. Private key encryption could prevent an attacker Eve from knowing what Bob bids, but if she could construct a message that contained one more than Bob's bid, then she could win the auction.

We say that an encryption scheme that prevents these kinds of attacks is *non-malleable*. In such a scheme, it is impossible for an adversary to output a ciphertext that corresponds to any function of a given encrypted message. Formally, we have the following definition:

▷**Definition 172.6 (Non-Malleable Encryption)** *Let the triple* $(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$

be a public key encryption scheme. Define the following experiment:

```
$\mathsf{NM}_b(\Pi, ma, n)$
  $k \leftarrow \mathsf{Gen}(1^n)$
  $m_0,\ m_1,\ state \leftarrow A^{O_1(k)}(1^n)$
  $c \leftarrow \mathsf{Enc}_k(m_b)$
  $c'_1, c'_2, c'_3, \ldots, c'_\ell \leftarrow A^{O_2(k)}(c, state)$
  $m'_i \leftarrow \begin{cases} \perp & \text{if } c_i = c \\ \mathsf{Dec}_k(c'_i) & \text{otherwise} \end{cases}$
Output $(m'_1, m'_2, \ldots, m'_\ell)$
```

Then (Gen, Enc, Dec) *is* non-malleable *if for every non-uniform p.p.t. $\mathcal{A}$, and for every non-uniform p.p.t. $\mathcal{D}$, there exists a negligible ϵ such that for all $m_0, m_1 \in \{0,1\}^n$,*

$$\Pr\left[\mathcal{D}(NM_0(\Pi, \mathcal{A}, n)) = 1\right] - \Pr\left[\mathcal{D}(NM_1(\Pi, \mathcal{A}, n)) = 1\right] \leq \epsilon(n)$$

One non-trivial aspect of this definition is the conversion to $\perp$ of queries that have already been made (step 4). Clearly without this, the definition would be trivially unsatisfiable, because the attacker could simply "forge" the encryptions that they have already seen by replaying them.

7.1.6 Relation-Based Non-Malleability

We chose this definition because it mirrors our definition of secrecy in a satisfying way. However, an earlier and arguably more natural definition can be given by formalizing the intuitive notion that the attacker cannot output an encryption of a message that is related to a given message. For example, we might consider the relation $R_{\text{next}}(x) = \{x+1\}$, or the relation $R_{\text{within-one}}(x) = \{x-1, x, x+1\}$. We want to ensure that the encryption of x does not help the attacker encrypt an element of $R(x)$. Formally:

▷**Definition 173.7 (Relation-Based Non-Malleable Encryption)** *An encryption scheme* (Gen, Enc, Dec) *is relation-based non-malleable if for every p.p.t. adversary $\mathcal{A}$ there exists a p.p.t. simulator $\mathcal{S}$ such that* *for all p.p.t.-recognizable relations R, there exists a negligible ϵ such that for all $m \in \mathcal{M}$ with $|m| = n$, and for all z, it holds that*

$$\left| \begin{array}{l} \Pr[NM(A(z), m) \in R(m)] \\ \quad - \Pr[k \leftarrow \mathsf{Gen}(1^n); c \leftarrow \mathcal{S}(1^n, z) : \mathsf{Dec}_k(c) \in R(m)] \end{array} \right| < \epsilon$$

where i ranges from 1 to a polynomial of n and NM is defined as above.

This definition is equivalent to the non-relational definition given above.

▷**Theorem 174.8** *Scheme* $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Gen})$ *is a non-malleable encryption scheme if and only if it is a relation-based non-malleable encryption scheme.*

Proof. $(\Rightarrow)$ Assume that the scheme is non-malleable by the first definition. For any given adversary $\mathcal{A}$, we need to produce a simulator $\mathcal{S}$ that hits any given relation R as often as $\mathcal{A}$ does. Let $\mathcal{S}$ be the machine that performs the first 3 steps of $NM(\mathcal{A}(z), m')$ and outputs the sequence of cyphertexts, and let $\mathcal{D}$ be the distinguisher for the relation R. Then

$$\begin{aligned} &\left| \begin{array}{l} \Pr[NM(\mathcal{A}(z), m) \in R(m)] - \\ \quad \Pr[k \leftarrow \mathsf{Gen}(1^n); c \leftarrow \mathcal{S}(1^n, z); m' = \mathsf{Dec}_k(c) : m' \in R(m)] \end{array} \right| \\ &= |\Pr[\mathcal{D}(NM(\mathcal{A}(z), m))] - \Pr[\mathcal{D}(NM(\mathcal{A}(z), m'))]| \leq \epsilon \end{aligned}$$

as required.
$(\Leftarrow)$ Assume that the scheme is relation-based non-malleable. Given an adversary $\mathcal{A}$, we know there exists a simulator $\mathcal{S}$ that outputs related encryptions as well as $\mathcal{A}$ does. The relation-based definition tells us that $NM(\mathcal{A}(z), m_0) \approx Dec(\mathcal{S}())$ and $Dec(\mathcal{S}()) \approx NM(\mathcal{A}(z), m_1)$. Thus, by the hybrid lemma, it follows that $NM(\mathcal{A}(z), m_0) \approx NM(\mathcal{A}(z), m_1)$ which is the first definition of non-malleability. □

7.1.7 Non-Malleability and Secrecy

Note that non-malleability is a distinct concept from secrecy. For example, one-time pad is perfectly secret, yet is not non-malleable (since one can easily produce the encryption of $a \oplus b$ give then encryption of a, for example). However, if we consider security under CCA2 attacks, then the two definitions coincide.

▷**Theorem 175.9** *An encryption scheme* (Enc, Dec, Gen) *is CCA2 secret if and only if it is CCA2 non-malleable*

Proof. (Sketch) If the scheme is not CCA2 non-malleable, then a CCA2 attacker can break secrecy by changing the provided encryption into a related encryption, using the decryption oracle on the related message, and then distinguishing the unencrypted related messages. Similarly, if the scheme is not CCA2 secret, then a CCA2 attacker can break non-malleability by simply decrypting the cyphertext, applying a function, and then re-encrypting the modified message. □

7.2 Composition of Zero-knowledge Proofs*

7.2.1 Sequential Composition

Whereas the definition of zero knowledge only talks about a *single* execution between a prover and a verifier, the definitions is in fact closed under sequential composition; that is, sequential repetitions of a ZK protocol results in a new protocol that still remains ZK.

▷**Theorem 175.1 (Sequential Composition)** *Let (P, V) be a perfect/computational zero-knowledge proof for the language L. Let $Q(n)$ be a polynomial, and let (P_Q, V_Q) be an interactive proof (argument) that on common input $x \in \{0,1\}^n$ proceeds in $Q(n)$ phases, each on them consisting of an execution of the interactive proof (P, V) on common input x (each time with independent random coins). Then (P_Q, V_Q) is an perfect/computational* ZK *interactive proof.*

Proof. (Sketch) Consider a malicious verifier V^{Q*}. Let

$$V^*(x, z, r, (\bar{m}_1, \ldots, \bar{m}_i))$$

denote the machine that runs $V^{Q*}(x, z)$ on input the random tape r and feeds it the messages $(\bar{m}_1, \ldots, \bar{m}_i)$ as part of the i first iterations of (P, V) and runs just as V^{Q*} during the $i+1$ iteration, and then halts. Let S denote the zero-knowledge simulator for V^*. Let $p(\cdot)$ be a polynomial bounding the running-time of V^{Q*}. Condsider now the simulator S^{Q*} that proceeds as follows on input x, z

- Pick a length $p(|x|)$ random string r.
- Next proceed as follows for $Q(|x|)$ iterations:
  - In iteration i, run $S(x, z||r||(\bar{m}_1, \ldots, \bar{m}_i))$ and let $\bar{m}_{i+1}$ denote the messages in the view output.

The linearity of expectations, the expected running-time of S^Q is polynomial (since the expected running-time of S is). A standard hybrid argument can be used to show that the output of S^Q is correctly distributed. □

7.2.2 Parallel/Concurrent Composition

Sequential composition is a very basic notion of compostion. An often more realistic scenario consider the execution of multiple protocols at the same time, with an arbitrary scheduling. As we show in this section, zero-knowledge is not closed under such "concurrent composition". In fact, it is not even closed under "parallel-composition" where all protocols executions start at the same time and are run in a lockstep fashion.

Consider the protocol (P, V) for proving $x \in L$, where P on input x, y and V on input x proceed as follows, and L is a language with a unique witness (for instance, L could be the language consisting of all elements in the range of a $1-1$ one-way function f, and the associated witness relation is $R_L(x) = \{y | f(y) = x\}$.

PROTOCOL 176.2: ZK PROTOCOL THAT IS NOT CONCURRENTLY SECURE

$P \to V$ P provides a zero-knowledge proof of knowledge of $x \in L$.

$P \leftarrow V$ V either "quits" or starts a zero-knowledge proof of knowledge $x \in L$.

$P \to V$ If V provides a convincing proof, P reveals the witness y.

It can be shown that the (P, V) is zero-knowledge; intuitively this follows from the fact that P only reveals y in case the verifier already knows the witness. Formally, this can be shown by "extracting" y from any verifier V^* that manages to convince P. More precisely, the simulator S first runs the simulator for the ZK proof in step 1; next, if V^* produces an accepting proof in step 2, S runs the extractor on V^* to extract a witness y' and finally feeds the witness to y'. Since by assumption L has a unique witness it follows that $y = y'$ and the simulation will be correctly distributed.

However, an adversary A that participates in two concurrent executions of (P, V), acting as a verifier in both executions, can easily get the witness y even if it did not know it before. A simply schedules the messages such that the zero-knowledge proof that the prover provides in the first execution is forwarded as the step 2 zero-knowledge proof (by the verifier) in the second execution; as such A convinces P in the second execution that it knows a witness y (although it is fact only is relaying messages from the the other prover, and in reality does not know y), and as a consequence P will reveal the witness to A.

The above protocol can be modified (by padding it with dummy messages) to also give an example of a zero-knowledge protocol that is not secure under even two parallel executions.

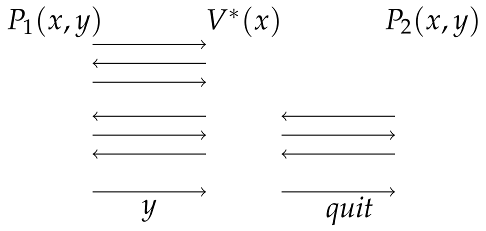

Figure 177.3: A Message Schedule which shows that protocol 176.2 does not concurrently compose. The Verifier feeds the prover messages from the second interaction with P_2 to the first interaction with prover P_1. It therefore convinces the first prover that it "knows" y, and therefore, P_1 sends y to V^*.

7.2.3 Witness Indistinguishability

- Definition
- WI closed under concurrent comp
- ZK implies WI

7.2.4 A Concurrent Identification Protocol

- y_1, y_2 is pk
- x_1, x_2 is sk
- WI POK that you know inverse of either y_1 or y_2.

7.3 Composition Beyond Zero-Knowledge Proofs

7.3.1 Non-malleable commitments

- Mention that standard commitment is malleable
- (could give construction based on adaptive OWP?)

Chapter 8

*More on Randomness and Pseudorandomness

8.1 A Negative Result for Learning

Consider a space $S \subseteq \{0,1\}^n$ and a family of concepts $\{C_i\}_{i \in I}$ such that $C_i \subseteq S$.

The Learning Question: For a random $i \in I$, given samples (x_j, b_j) such that $x_j \in S$ and $b_j = 1$ iff $x_j \in C_i$, determine for a bit string x if $x \in C_i$. The existence of PRFs shows that there are concepts that can not be learned.

▷**Theorem 179.1** *There exists a p.p.t. decidable concept that cannot be learned.*

Proof sketch.

$$\begin{aligned} S &= \{0,1\}^n \\ C_i &= \{x \mid f_i(x)_{|1} = 1\} \qquad f_i(x)_{|1} \text{ is the first bit of } f_i(x) \\ I &= \{0,1\}^n \end{aligned}$$

No (n.u.) p.p.t. can predict whether a new sample x is in C_i better than $\frac{1}{2} + \epsilon$. □

8.2 Derandomization

Traditional decision problems do not need randomness; a randomized machine can be replaced by a deterministic machine that tries all finite random tapes. In fact, we can do better if we make some cryptographic assumptions. For example:

▷**Theorem 180.1** *If pseudo-random generators (PRG) exist, then for every constant $\varepsilon > 0$, **BPP** $\subseteq$ **DTIME**(2^{n^ε}).*

Proof. where **DTIME**$(t(n))$ denotes the set of all languages that can be decided by deterministic machines with running-time bounded by $O(t(n))$.

Given a language $L \in$ **BPP**, let M be a p.p.t.Turing machine that decides L with probability at least 2/3. Since the running time of M is bounded by n^c for some constant c, M uses at most n^c bits of the random tape. Note that we can trivially de-randomize M by deterministically trying out all 2^{n^c} possible random tapes, but such a deterministic machine will take more time than 2^{n^ε}.

Instead, given ε, let $g : \{0,1\}^{n^{\varepsilon/2}} \to \{0,1\}^{n^c}$ be a PRG (with polynomial expansion factor $n^{c-\varepsilon/2}$). Consider a p.p.t.machine M' that does the following given input x:

1. Read $\varepsilon/2$ bits from the random tape, and apply g to generate n^c pseudo-random bits.

2. Simulate and output the answer of M using these pseudo-random bits.

M' must also decide L with probability negligibly close to 2/3; otherwise, M would be a p.p.t.distinguisher that can distinguish between uniform randomness and the output of g.

Since M' only uses $n^{\varepsilon/2}$ random bits, a deterministic machine that simulates M' on all possible random tapes will take time

$$2^{n^{\varepsilon/2}} \cdot \text{poly}(n) \in O(2^{n^\varepsilon})$$

□

Remark: We can strengthen the definition of a PRG to require that the output of a PRG be indistinguishable from a uniformly random string, even when the distinguisher can run in subexponential time (that is, the distinguisher can run in time $t(n)$ where $t(n) \in O(2^{n^\varepsilon})$ for all $\varepsilon > 0$). With this stronger assumption, we can show that $\mathbf{BPP} \subseteq \mathbf{DTIME}(2^{\mathrm{poly}(\log n)})$, the class of languages that can be decided in *quasi-polynomial* time.

However, for cryptographic primitives, we have seen that randomness is actually required. For example, any deterministic public key encryption scheme must be insecure. But how do we get randomness in the real world? What if we only have access to "impure" randomness?

8.3 Imperfect Randomness and Extractors

In this section we discuss models of imperfect randomness, and how to extract truly random strings from imperfect random sources with *deterministic extractors*.

8.3.1 Extractors

Intuitively, an extractor should be an efficient and deterministic machine that produces truly random bits, given a sample from an imperfect source of randomness. In fact, sometimes we may be satisfied with just "almost random bits", which can be formalized with the notion of ε-closeness.

▷**Definition 181.1 (ε-closeness)** *Two distributions X and Y are ε-*close, *written $X \approx_\varepsilon Y$, if for every (deterministic) distinguisher D (with no time bound),*

$$|\Pr[x \leftarrow X : D(x) = 1] - \Pr[y \leftarrow Y : D(y) = 1]| \leq \varepsilon$$

▷**Definition 181.2 (ε-extractors)** *Let C be a set of distributions over $\{0,1\}^n$. An m*-bit ε-extractor *for C is a deterministic function* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ *that satisfies the following:*

$$\forall X \in C,\ \{x \leftarrow X : \mathrm{Ext}(x)\} \approx_\varepsilon U_m$$

where U_m is the uniform distribution over $\{0,1\}^m$.

8.3.2 Imperfect Randomness

An obvious example of imperfect randomness is to repeatedly toss a biased coin; every bit in the string would be biased in the same manner (i.e. the bits are independently and identically distributed). Van Neumann showed that the following algorithm is a 0-extractor (i.e. algorithm produces truly random bits): Toss the biased coin twice. Output 0 if the result was 01, output 1 if the result was 10, and repeat the experiment otherwise.

A more exotic example of imperfect randomness is to toss a sequence of different biased coins; every bit in the string would still be independent, but not biased the same way. We do not know any 0-extractor in this case. However, we can get a ε-extractor by tossing a sufficient large number of coins at once and outputting the XOR of the results.

More generally, one can consider distributions of bit strings where different bits are not even independent (e.g. bursty errors in nature). Given an imperfect source, we would like to have a measure of its "amount of randomness". We first turn to the notion of *entropy* in physics:

▷**Definition 182.3 (Entropy)** *Given a distribution X, the* entropy *of X, denoted by $H(x)$ is defined as follows:*

$$\begin{aligned} H(X) &= \mathbb{E}\left[x \leftarrow X : \log\left(\frac{1}{\Pr[X=x]}\right)\right] \\ &= \sum_x \Pr[X=x]\log\left(\frac{1}{\Pr[X=x]}\right) \end{aligned}$$

When the base of the logarithm is 2, $H(x)$ is the Shannon entropy of X.

Intuitively, Shannon entropy measures how many truly random bits are "hidden" in X. For example, if X is the uniform distribution over $\{0,1\}^n$, X has Shannon entropy

$$H(X) = \sum_{x\in\{0,1\}^n} \Pr[X=x]\log_2\left(\frac{1}{\Pr[X=x]}\right) = 2^n(2^{-n}\cdot n) = n$$

As we will soon see, however, a source with high Shannon entropy can be horrible for extractors. For example, consider X

defined as follows:

$$X = \begin{cases} 0^n & \text{w.p. } 0.99 \\ \text{uniformly random element in } \{0,1\}^n & \text{w.p. } 0.01 \end{cases}$$

Then, $H(X) \approx 0.01n$. However, an extractor that samples an instance from X will see 0^n most of the time, and cannot hope to generate even just one random bit[1]. Therefore, we need a stronger notion of randomness.

▷**Definition 183.4 (Min Entropy)** *The* min entropy *of a probability distribution X, denoted by $H_\infty(x)$, is defined as follows:*

$$H_\infty(X) = \min_x \log_2 \left(\frac{1}{\Pr[X = x]} \right)$$

Equivalently,

$$H_\infty(X) \geq k \Leftrightarrow \forall x,\ \Pr[X = x] \leq 2^{-k}$$

▷**Definition 183.5 (k-source)** *A probability distribution X is called a k-*source *if $H_\infty(X) \geq k$. If additionally X is the uniform distribution on 2^k distinct elements, we say X is a k-*flat source.

Even with this stronger sense of entropy, however, extraction is not always possible.

▷**Theorem 183.6** *Let C be the set of all efficiently computable $(n-2)$-sources on $\{0,1\}^n$. Then, there are no 1-bit $1/4$-extractors for C.*

Proof. Suppose the contrary that Ext is a $1/4$-extractor for C. Consider the distribution X generated as follows:

1. Sample $x \leftarrow U_n$. If $\text{Ext}(x) = 1$, output x. Otherwise repeat.
2. After 10 iterations with no output, give up and output a random $x \leftarrow U_n$.

[1] A possible fix is to sample X many times. However, we restrict ourselves to one sample only motivated by the fact that some random sources in nature can not be independently sampled twice. E.g. the sky in the morning is not independent from the sky in the afternoon.

Since $U_n \in C$ and Ext is a 1/4-extractor, we have

$$\Pr[x \leftarrow U_n : \text{Ext}(x) = 1] \geq 1/2 - 1/4 = 1/4$$

which implies that $|\left\{x \in \{0,1\}^n : \text{Ext}(x) = 1\right\}| \geq (1/4)2^n = 2^{n-2}$. We can then characterize X as follows:

$$X = \begin{cases} U_n \text{ w.p.} \leq \left(\frac{3}{4}\right)^{10} \\ \text{uniform sample from } \left\{x \in \{0,1\}^n, \text{Ext}(x) = 1\right\} \text{ o.w.} \end{cases}$$

Since $|\left\{x \in \{0,1\}^n, \text{Ext}(x) = 1\right\}| \geq 2^{n-2}$, both cases above are $(n-2)$-sources. This makes X a $(n-2)$-source. Moreover, X is computable in polynomial time since Ext is. This establishes $X \in C$.

On the other hand,

$$\Pr[x \in X : \text{Ext}(x) = 1] \geq 1 - \left(\frac{3}{4}\right)^{10} > 0.9$$

and so $\{x \in X : \text{Ext}(x)\}$ is definite not 1/4-close to U_1, giving us the contradiction. □

8.3.3 Left-over hash lemma

Bibliography

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160(2):781–793, 2004.

[BBFK05] Friedrich Bahr, Michael Böhm, Jens Franke, and Thorsten Kleinjung. Announcement of the factorization of rsa-200, May 2005. http://www.crypto-world.com/FactorAnnouncements.html.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms (3rd Edition)*. MIT Press, 2009.

[EUL63] Leonhard Euler. Theoremata arithmetica nova methodo demonstrata. In *Novi Commentarii academiae scientiarum Petropolitanae 8*, pages 74–104. –, 1763.

[KAF$^+$10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit rsa modulus. eprint.iacr.org/2010/006.pdf, January 2010.

[KNU81] Donald E. Knuth. *The Art of Computer Programming: Seminumerical algorithms*, volume 2. Addison-Wesley, 2nd edition edition, 1981.

[MIL76] Gary Miller. Riemanns hypothesis and tests for primality. *J. Comput. System Sci*, 13(3):300–317, 1976.

[RAB80] Michael Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1):128–138, 1980.

[SHA49] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

Appendix A

Background Concepts

Basic Probability

- Events A and B are said to be *independent* if

$$Pr[A \cap B] = Pr[A] \cdot Pr[B]$$

- The *conditional probability* of event A given event B, written as $\Pr[A \mid B]$ is defined as

$$\Pr[A \mid B] = \frac{\Pr[A \cap B]}{\Pr[B]}$$

- *Bayes theorem* relates the $\Pr[A \mid B]$ with $\Pr[B \mid A]$ as follows:

$$\Pr[A \mid B] = \frac{\Pr[B \mid A] \Pr[A]}{\Pr[B]}$$

- Events $A_1, A_2, \ldots, A_n$ are said to be *pairwise independent* if for every i and every $j \neq i$, A_i and A_j are independent.

- *Union Bound:* Let $A_1, A_2, \ldots, A_n$ be events. Then,

$$\Pr[A_1 \cup A_2 \cup \ldots \cup A_n] \leq \Pr[A_1] + \Pr[A_2] + \ldots + \Pr[A_n]$$

- Let X be a random variable with range Ω. The *expectation* of X is the value:

$$E[X] = \sum_{x \in \Omega} x \Pr[X = x]$$

The *variance* is given by,

$$Var[X] = E[X^2] - (E[X])^2$$

- Let $X_1, X_2, \ldots, X_n$ be random variables. Then,

$$E[X_1 + X_2 + \cdots + X_n] = E[X_1] + E[X_2] + \cdots + E[X_n]$$

- If X and Y are *independent random variables*, then

$$E[XY] = E[X] \cdot E[Y]$$
$$Var[X+Y] = Var[X] + Var[Y]$$

Markov's Inequality

If X is a positive random variable with expectation $E(X)$ and $a > 0$, then

$$\Pr[X \geq a] \leq \frac{E(X)}{a}$$

Chebyshev's Inequality

Let X be a random variable with expectation $E(X)$ and variance σ^2, then for any $k > 0$,

$$\Pr[|X - E(X)| \geq k] \leq \frac{\sigma^2}{k^2}$$

Chernoff's inequality

▷**Theorem 188.7** *Let $X_1, X_2, \ldots, X_n$ denote independent random variables, such that for all i, $E(X_i) = \mu$ and $|X_i| \leq 1$.*

$$\Pr\left[\left|\sum X_i - \mu n\right| \geq \epsilon\right] \leq 2^{-\epsilon^2 n}$$

The constants in this statement have been specifically chosen for simplicity; they can be further optimized for tighter analysis.

A useful application of this inequality is the *Majority voting* lemma. Assume you can get independent and identically distributed, but *biased*, samples of a bit b; that is, these samples are correct only with probability $\frac{1}{2} + \frac{1}{\text{poly}(n)}$. Then, given poly$(n)$ samples, compute the most frequent value b' of the samples; it holds with high probability that $b = b'$.

▷**Lemma 189.8 (Majority vote)** *Let $b \in \{0,1\}$ be a bit and let $X_1, \ldots, X_\ell$ denote independent random variables such that* $\Pr[X_i = b] \geq \frac{1}{2} + \frac{1}{p(n)}$ *for some polynomial p. Then if $\ell > p(n)^2$,*

$$\Pr[\text{majority}(X_1, \ldots, X_\ell) = b] > 1 - 2^{??}$$

Proof. Without loss of generality, assume that $b = 1$. (Similar analysis will apply in the case $b = 0$.) In this case, $\mu = E[X_i] = \frac{1}{2} + \frac{1}{p(n)}$, and so $\mu\ell = \ell(\frac{1}{2} + \frac{1}{p(n)}) > \ell/2 + p(n)$. In order for the majority procedure to err, less than $\ell/2$ of the samples must agree with b; i.e. $\sum X_i < \ell/2$. Applying the Chernoff bound, we have that ...

□

Pairwise-independent sampling inequality

Let $X_1, X_2, \ldots, X_n$ denote pair-wise independent random variables, such that for all i, $E(X_i) = \mu$ and $|X_i| \leq 1$.

$$\Pr\left[\left|\frac{\sum X_i}{n} - \mu\right| \geq \epsilon\right] \leq \frac{1 - \mu^2}{n\epsilon^2}$$

Note that this is a Chernoff like bound when the random variables are only pairwise independent. The inequality follows as a corollary of Chebyshev's inequality.

Cauchy-Schwarz Inequality

In this course, we will only need Cauchy's version of this inequality from 1821 for the case of real numbers. This inequality states that

▷**Theorem 189.9 (Cauchy-Schwarz)** *For real numbers x_i and y_i,*

$$\left(\sum_i^n x_i y_i\right)^2 \leq \sum_i^n x_i^2 \cdot \sum_i^n y_i^2$$

Appendix B

Basic Complexity Classes

We recall the definitions of the basic complexity classes **DP**, **NP** and **BPP**.

The Complexity Class DP. We start by recalling the definition of the class **DP**, i.e., the class of languages that can be decided in (deterministic) polynomial-time.

▷**Definition 191.10 (Complexity Class DP)** *A language L is recognizable in (deterministic) polynomial-time if there exists a deterministic polynomial-time algorithm M such that $M(x) = 1$ if and only if $x \in L$. **DP** is the class of languages recognizable in polynomial time.*

The Complexity Class NP. We recall the class **NP**, i.e., the class of languages for which there exists a proof of membership that can be verified in polynomial-time.

▷**Definition 191.11 (Complexity Class NP)** *A language L is in **NP** if there exists a Boolean relation $R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ and a polynomial $p(\cdot)$ such that R_L is recognizable in polynomial-time, and $x \in L$ if and only if there exists a string $y \in \{0,1\}^*$ such that $|y| \leq p(|x|)$ and $(x, y) \in R_L$.*

The relation R_L is called a *witness relation* for L. We say that y is a witness for the membership $x \in L$ if $(x, y) \in R_L$. We will also let $R_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e.,

$$R_L(x) = \{y : (x, y) \in L\}$$

We let co-**NP** denote the complement of the class **NP**, i.e., a language L is in co-**NP** if the complement to L is in **NP**.

The Complexity Class BPP. The class **BPP** contains the languages that can be decided in *probabilistic* polynomial-time (with two-sided error).

▷**Definition 192.12 (Complexity Class BPP)** *A language L is recognizable in probabilistic polynomial-time if there exists a probabilistic polynomial-time algorithm M such that*

- $\forall x \in L, \Pr[M(x) = 1] \geq 2/3$
- $\forall x \notin L, \Pr[M(x) = 0] \geq 2/3$

***BPP** is the class of languages recognizable in probabilistic polynomial time.*