Hardcore bits, coin flipping over the phone, and commitment schemes

Noah Stephens-Davidowitz

May 29, 2023

1 Preview: The journey from OWF to SKE

So far, we have seen the definition of semantically secure encryption and one-way functions, and we have seen a few examples of functions that we believe are one way—hopefully enough examples to convince you that one-way functions probably exist. The goal of the next few lectures will be to prove that the existence of one-way functions implies the existence of semantically secure encryption.

This is a fundamental result in the field of cryptography, proven in a long sequence of works (and, for us, in a long sequence of lectures). The proof is *quite* beautiful, and it has the huge benefit of introducing us to some of the key ideas that appear in Minicrypt and throughout cryptography more generally. In particular, we will see the concept of pseudorandomness, and the proof technique of a hybrid argument. The result itself is also just very surprising—one-way functions seem like quite weak objects compared to secret-key encryption.

An unfortunate aspect of the proof is that the first step, the celebrated Goldreich-Levin theorem, is probably the most difficult. (In fact, what we *really* need is a strengthening of Goldreich-Levin known as the HILL theorem [HILL99], which is far too difficult for this course. Goldreich-Levin is hard enough :).)

In this lecture, we will see the concept of a hardcore predicate (which is often informally referred to as a hardcore bit). We will see two important *direct* applications of hardcore predicates here. But, the biggest application of hardcore predicates—as part of the bridge from one-way functions to secret-key encryption—will have to wait.

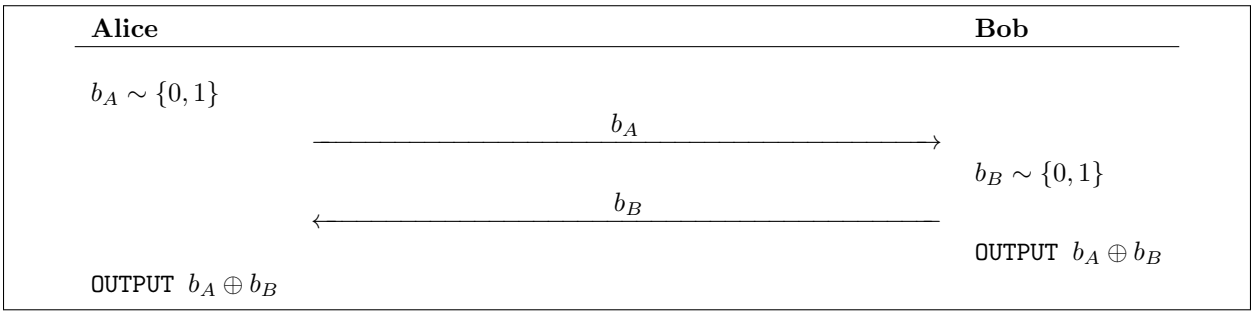
2 Coin flipping over the phone

The idea of coin flipping over the phone was coined by Manuel Blum [Blu83]. (He actually called his protocol for coin flipping "a protocol for solving impossible problems," acknowledging that the problem that he solved seems paradoxical at first. You'll see what he meant!) Blum imagined a rather unfortunate situation in which Alice and Bob (who were getting along so well in our first lecture!) are getting a divorce. They've already separated—moved to different sides of the country—and now they're trying to decide who gets which of their joint assets. In particular, they both love their car, and neither one is willing to give it up.

Since they can't just cut the car in half, Alice and Bob decide that the only fair thing to do is to flip a coin. If the coin comes up heads, Alice gets the car. If it's tails, then Bob gets it.

But, they're across the country from each other. So, they aren't able to physically flip a coin together (and it's 1983, so they can't flip a coin on video). Of course, Alice could just flip a coin by herself and tell Bob the result, but Bob simply isn't willing to trust Alice quite that much (nor is Alice willing to trust Bob). So, what do they do? Slightly more formally: what protocol can Alice and Bob engage in to jointly agree on a coin flip without allowing one of them to heavily bias the outcome?

Here's a *bad* idea! Suppose Alice and Bob engage in the following protocol.



This protocol is just as bad as the protocol in which Alice flips a coin and tells Bob the result. In the "only Alice flips" protocol, if Alice is malicious, she can directly choose whatever output she likes. In the above slightly more complicated protocol, if Bob is malicious, then he can pick whatever output $b^* \in \{0,1\}$ that he prefers by simply choosing $b_B := b^* \oplus b_A$.

In both protocols, the issue is that "whoever speaks last can choose the outcome." This might seem inevitable for any protocol, but if we place computational bounds on Alice and Bob, then it is possible to guarantee that neither Alice nor Bob can have much control over the outcome—assuming, of course, that one-way functions exist. (Actually, we will only need to place a computational bound on one party, which will be Bob in our case. It's a good exercise to convince yourself that it's definitely impossible to jointly flip an unbiased coin if both Alice and Bob are computationally unbounded.)

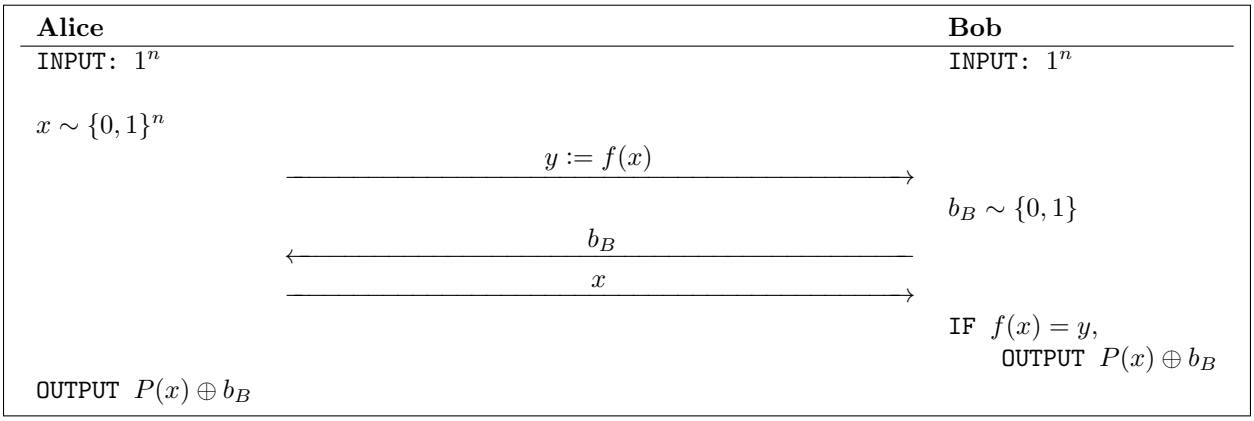
Here is the rough idea of the protocol. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a one-way function. In fact, for simplicity, we will make one other assumption about f; it should be injective. In other words, for every $y \in \{0,1\}^*$, there should be at most one $x \in \{0,1\}^*$ with $f(x) = y$. (This is a very big assumption. It is not *strictly* necessary, but it will make our task *a lot* easier. We will not see how to remove this assumption in this course.) For example, the one-way functions that we constructed using the discrete logarithm are injective (and injective one-way functions can also be constructed based on factoring).

Then, instead of just sending some bit b_A to Bob, Alice will sample $x \sim \{0,1\}^n$ and send $y := f(x)$ to Bob. Intuitively, we think that "Bob has no idea what x is." Bob will then send a random bit $b_B \sim \{0,1\}$ to Alice as normal. Finally, Alice sends x to Bob, Bob checks that $y := f(x)$, and Alice and Bob will use b_B and x together to compute y. (If Bob's check fails, then let's just assume that Bob gets the car. I.e., if Alice cheats in some way and Bob catches her, then we assume that she forfeits her right to the car.) Formally, they will output $P(x) \oplus b_B$, where $P : \{0,1\}^* \to \{0,1\}$ is some efficiently computable predicate. (A *predicate* is just a fancy name for a function whose output is a single bit.)

Intuitively, as long as "Bob cannot determine $P(x)$ from $f(x)$," he should not be able to sig-

nificantly bias the output $P(x) \oplus b_B$. And, since f is injective, there is only one choice of x that Alice can send with $f(x) = y$, so Alice has no control at all over the output.

Here's the protocol written formally.



The security definition (which we will not bother to write formally in this lecture) for this primitive is as follows: for any PPT adversary playing the role of Bob there is a negligible $\varepsilon(n)$, such that for any bit b^*,

$$\Pr[P(x) \oplus b_B = b^*] \leq 1/2 + \varepsilon(n) \ .$$

In the next section, we will see a necessary and sufficient condition on the function $P(x)$ to achieve this.

3 Hardcore predicates

Intuitively, in order to prevent Bob from controlling the output bit $P(x) \oplus b_B$, it should be the case that $P(x)$ is difficult for Bob to compute given $y = f(x)$. In particular, if $P(x)$ were easy to compute, then Bob could set $b_B = P(x) \oplus b^*$ for whatever b^* he prefers, and therefore control the output bit. In fact, even if Bob could guess $P(x)$ with probability significantly larger than $1/2$, then Bob could use this same strategy to bias the output bit. So, we actually want to make sure that $P(x)$ cannot be guessed with probability much greater than $1/2$, given $f(x)$. (You should be able to convince yourself that, with appropriately formal definitions, this condition is both necessary and sufficient.)

Definition 3.1. *We say that a predicate $P : \{0,1\}^* \to \{0,1\}$ is a* hardcore predicate for a function $f : \{0,1\}^* \to \{0,1\}^*$ *if it satisfies the following two properties.*

- **(Easy to compute given x.)** *There exists a PPT $\mathcal{B}$ such that $\mathcal{B}(x) = P(x)$ for all x.*
- **(Hard to guess given $f(x)$.)** *For any PPT $\mathcal{A}$, there exists a negligible function $\varepsilon(n)$ such that*

$$\Pr_{x \sim \{0,1\}^n}[b \leftarrow \mathcal{A}(1^n, f(x)),\ b = P(x)] \leq 1/2 + \varepsilon(n) \ .$$

For example, it is known that, if (G_n, g_n) are a family of groups of order p_n over which the discrete logarithm is hard, then the predicate

$$P(k) := \begin{cases} 1 & k > p_n/2 \\ 0 & k \leq p_n/2 \end{cases}$$

is hardcore for the function $f(k) = g_n^k$. In other words, given g_n^k, it is difficult to determine whether $k > p_n/2$. This can be proven relatively easily. (I sometimes give it for homework.)

3.1 Aside: Mental poker

Actually, our coin-flipping protocol allows Alice and Bob to do a little bit more than flip coins. They can play poker over the telephone, which is often referred to as mental poker—i.e., a game of poker in which "the cards are only in Alice and Bob's minds." (A formal version of this idea was first developed in [SRA79].)

Of course, the difficulty with playing poker over the telephone is that there is nobody to deal the cards. Alice could let Bob deal the cards and tell her what hand she got, but then Bob would know her hand, which makes poker rather silly!

Alternatively, Alice could deal herself a hand and Bob could deal himself a hand. Then, neither player will know the other player's hand. This protocol has the unfortunate issue that Alice and Bob might both hold the same card, which does not happen when people play with a real deck of cards, but we will ignore that problem.[1] A more serious problem is that Alice and Bob could simply lie! Just like in the coin-flipping story above, Alice is not willing to trust Bob to deal cards honestly to himself, nor is Bob willing to trust Alice.

Notice that our coin-flipping protocol actually solves this problem too! The only new idea that is necessary is to save the last message for later. Specifically, to deal Alice's cards, Alice and Bob can engage in their coin-flipping protocol repeatedly, enough times to flip enough coins to deal Alice the right number of cards for the game that they are playing. However, in each of these protocols, Alice will *not* send her final message in which she reveals x until later. Specifically, she will not send this message until it is time to reveal her cards. To deal Bob's cards, the two parties do the same thing with their roles reversed.

In this way, Alice and Bob can together deal cards in such a way that neither party can bias the outcome (by a non-negligible amount), only Alice knows her cards, and only Bob knows his (assuming both parties are PPT). And, if they get to the end of a hand, they can jointly reveal their cards. (This was the "impossible problem" that Blum referred to in the title of his paper [Blu83].)

4 Commitment schemes (and magicians)

So, hardcore predicates are useful for coin flipping over the telephone. They are *also* very useful for constructing *commitment schemes*. In fact, our application to coin flipping was really just a commitment scheme in disguise.

The story behind a commitment scheme is as follows. Alice wants to *commit* to some plaintext m so that later on, she can reveal m to Bob (called *opening* the commitment) in such a way that

[1] This can be dealt with elegantly, but an inelegant solution is for Alice and Bob to simply check after the fact whether they both held the same card. If they did, whatever happened in that hand should be undone. E.g., if Alice won \$100 from Bob when they both held the ace of spades, then Alice may not keep those \$100.

(1) Bob cannot determine what m is before Alice reveals it to him; and (2) Alice cannot change her mind about m after the fact—i.e., if Alice claims to have committed to some plaintext m, Bob can be sure that she really did.

Here is the formal definition. Notice that in this definition, we explicitly write the random coins of the algorithm Com. I.e., a randomized algorithm can be thought of as an algorithm that takes two things as input: the first is its "actual input," and the second is its random coins r. We use the notation $\mathsf{Com}(m; r)$, where the semicolon is meant to separate the two kinds of input. Conveniently, a randomized algorithm Com can be thought of as a *function* (i.e., something with a fixed output for every input) $\mathsf{Com}(m; r)$ when we treat the random coins as part of the input.[2]

We will show a commitment scheme on one bit. This is sufficient for our application, and it is easy to generalize it to larger plaintext spaces by simply concatenating many commitments.

Definition 4.1. *A* commitment scheme *is an efficient algorithm* $\mathsf{Com} : \{0,1\} \times \{0,1\}^* \to \{0,1\}^*$ *that takes as input (1) a plaintext bit* $b \in \{0,1\}$*; and (2) random coins* $r \in \{0,1\}^*$ *with the following two properties.*

- **(Perfectly binding.)** *For all* $r, r' \in \{0,1\}^*$, $\mathsf{Com}(0; r) \neq \mathsf{Com}(1; r')$.
- **(Computationally hiding.)** *For any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\varepsilon(n)$ *such that for all* n*,*

$$\Pr_{b \sim \{0,1\}, r \sim \{0,1\}^n}[b' \leftarrow \mathcal{A}(1^n, \mathsf{Com}(b; r)),\ b' = b] \leq 1/2 + \varepsilon(n)\ .$$

Then, to *commit* to a bit $b \in \{0,1\}$, Alice can do the following. She samples $r \sim \{0,1\}^n$ and sends the *commitment* $y := \mathsf{Com}(b; r)$ to Bob. Notice that the computationally hiding property guarantees that Bob cannot guess Alice's bit (with non-negligible advantage, assuming that Bob is PPT). If Alice later wishes to *open* (or reveal) the bit b to Bob, she can simply send Bob (b, r). Bob can then himself check that $y = \mathsf{Com}(b, r)$. Also notice that the binding property guarantees that Alice cannot commit to b and then later open her commitment to $1 - b$ because each commitment y can be opened to at most one bit b. I.e., the binding property *binds* (or commits) Alice to the bit b.

Indeed, these two properties together are exactly what we need to make a secure coin-flipping protocol. Specifically, to flip a coin, Alice chooses a random bit b_A and sends Bob a *commitment* to b_A. Bob then sends b_B to Alice; Alice opens her commitment to b_A; and they agree on the bit $b_A \oplus b_B$. The binding property guarantees that Alice has no control over the outcome, and the hiding property guarantees that Bob cannot bias the outcome by more than a negligible amount (assuming that he is PPT). (These statements are informal because I have not defined coin flipping formally. But, they can be made formal.)

Different notions of hiding and binding. The binding property above can be relaxed to a notion of *computationally binding*. This instead says that it is computationally difficult to find r, r' such that $\mathsf{Com}(0; r) = \mathsf{Com}(1; r')$. (I.e., no PPT adversary can accomplish this with non-negligible probability.) Since we almost always assume that all parties are PPT, and since we usually don't

[2] Formally, Com should also take the security parameter 1^n as input. In particular, Com should flip more coins if the security parameter is larger. Here, we simply assume that $|r| = n$ to avoid what would otherwise be rather clunky notation.

care about events that happen with negligible probability, this is essentially just as good. But, it is simpler for our purposes to work with this stronger *perfect* binding property.

There is also a notion of *perfectly hiding*. A perfectly hiding commitment scheme has the property that for every y, $\Pr_r[\mathsf{Com}(0;r) = y] = \Pr_r[\mathsf{Com}(1;r) = y]$. Notice that a scheme cannot be simultaneously perfectly hiding and perfectly binding. But, they can be computationally hiding and perfectly binding, or perfectly hiding and computationally binding. (Or, at least, this is true under reasonable complexity-theoretic assumptions.)

4.1 Aside: Commitment schemes are useful

Commitment schemes are useful for more than just coin-flipping protocols. We will see rather magical uses of commitment schemes for zero-knowledge proofs soon, and much later, for multi-party computation (which generalizes mental poker). But, notice that commitment schemes are also just useful for doing exactly what they claim to do: allowing someone to provably commit to something.

For example, suppose that Alice has invented something and wishes to talk to Bob, a patent lawyer. But, her invention is so valuable, that she is worried that Bob will steal her idea and claim that he thought of it first. To avoid this, Alice can create a commitment to a description m of her invention (one bit at a time, if we use the one-bit commitment schemes discussed above). She can then post this publicly somewhere, e.g., to Twitter, before talking to Bob. Now, if Bob later claims that the idea was his, Alice can open her commitment to prove that she knew about the idea before Bob. The binding property of the commitment scheme proves that what she posted to Twitter really was a commitment to m. On the other hand, the hiding property guarantees that nobody will be able to figure out what she posted before she posted it.

(Occasionally, people do in fact post what looks like a random hexadecimal string on Twitter, and often they are in fact committing to something. E.g., the controversial organization WikiLeaks has done so many times: https://twitter.com/wikileaks/status/787781519951720449?s=20. Presumably, they are doing this so that they can prove in the future that they knew something much earlier, without having to reveal what they know.)

4.2 Aside: Magicians break binding

There's a particular kind of magician (called a mentalist) whose performances tend to go as follows. First, the magician places some message in a sealed container—like an envelope or a box—apparently committing herself to whatever message she has placed in the container. Then, the magician asks someone in the audience to name something—say a number. Then, the magician opens the container to reveal that she had apparently written a prediction of what the audience member said before the audience member said it!

With these acts, magicians prove that one of two things is true. Either the magician can predict the future, or she has broken the "binding property" of the container, by revealing a different message than the one that she apparently committed to (or the audience member is colluding with the magician—but this is lame).

4.3 Aside: Magicians break hiding

Another trick that mentalists like to perform goes as follows. They first ask an audience member to write a number on a piece of paper or whatever and seal it somewhere in an envelope or whatever, apparently concealing the audience member's message. The magician then prances around the stage for a while and maybe holds the sealed envelope up to her head and pretends to concentrate really hard. Eventually, she declares that she knows what is in the envelope and says so to the audience. She then opens the envelope and shows the audience that she is in fact right!

With these acts, magicians prove that the container is not actually hiding (or that they have colluded with the audience member).

4.4 Commitment schemes from hardcore predicates

Finally, we show how to build a commitment scheme from an injective function with a hardcore predicate. The construction itself is a very elegant variant of the one-time pad, and it looks suspiciously like an encryption scheme. In general, I find it helpful to think of commitment schemes as something like "encryption schemes without decryption algorithms," though this is certainly not a formal characterization. The encryption scheme that we will eventually build will actually look quite similar to the construction below (but it *will* have a decryption algorithm!).

Theorem 4.2. *If there exists an injective function $f : \{0,1\}^* \to \{0,1\}^*$ with a hardcore predicate $P : \{0,1\}^* \to \{0,1\}$, then there exists a perfectly binding and computationally hiding commitment scheme.*

Proof. Let $\mathsf{Com}(b; r) := (f(r), P(r) \oplus b)$. In other words, the Com function "uses the predicate $P(r)$ as a one-time pad to hide b." The intuition for why this works is that "$P(r)$ is indistinguishable from a random bit, so it's just as good as a one-time pad."

We need to show that this is perfectly binding and computationally hiding. Indeed, the perfectly binding property follows immediately from the fact that f is injective. Pedantically, if $(f(r), P(r)) = (f(r'), P(r') \oplus 1)$ (i.e., if the scheme is not perfectly binding), then notice that we must have $r \neq r'$ (since $P(r) \neq P(r')$) and $f(r) = f(r')$, which contradicts the assumption that f is injective.

To show that the scheme is computationally hiding, consider some PPT adversary $\mathcal{A}$ with advantage

$$\varepsilon(n) := \Pr_{r \sim \{0,1\}^n, b \sim \{0,1\}}[b' \leftarrow \mathcal{A}(1^n, (f(r), b \oplus P(r))),\ b = b'] - 1/2\ .$$

in the commitment hiding game. We construct an adversary $\mathcal{A}'$ in the hardcore predicate game as follows. $\mathcal{A}'$ takes as input 1^n and $y := f(r)$, where $r \sim \{0,1\}^n$. $\mathcal{A}'$ then samples $b^* \sim \{0,1\}$ and runs $\mathcal{A}$ on input $(1^n, y, b^*)$, receiving as output some bit b'. Finally, $\mathcal{A}'$ outputs $b^* \oplus b'$.

Clearly, $\mathcal{A}'$ is PPT if $\mathcal{A}$ is PPT. So, it suffices to show that $\Pr[P(r) = b^* \oplus b'] = 1/2 + \varepsilon(n)$, which immediately implies that $\varepsilon(n)$ must be negligible (by the security of the hardcore predicate), which is what we wish to prove.

To that end, notice that the input $(1^n, y, b^*)$ that $\mathcal{A}'$ feeds to $\mathcal{A}$ has the exact same distribution as the input to $\mathcal{A}$ in the commitment hiding game. This is because the last bit of the input $b \oplus P(r)$ in the commitment game is uniformly random and independent of everything else, since b is sampled

uniformly at random and independent of everything else.[3] Therefore

$$\Pr[b^* = b' \oplus P(r)] = 1/2 + \varepsilon(n) \ ,$$

and the result follows. □

5 Committing to many-bit messages and our first *hybrid* argument

Above, we formally defined a commitment scheme for *one-bit* plaintexts $m \in \{0,1\}$. But, many of the applications that I described above actually required commitment of significantly longer plaintexts $m \in \{0,1\}^\ell$ for large ℓ—e.g., Alice might want to commit to a whole description of an invention, and she certainly might want to commit to more than a single bit!

Intuitively, if you want to commit to a many-bit message $m = (m_1, \ldots, m_\ell) \in \{0,1\}^\ell$, you should just commit to m_1, then m_2, then m_3, etc, using fresh independent randomness each time.

But, how do we know that this actually works? We need to prove it. Let's first generalize the definition of commitment from before to handle many messages, and then prove that we can achieve this new definition by just committing to a longer plaintext one bit at a time. Our proof will use a *hybrid argument*, which is a very strong proof technique that we will use *many* times in this class.

Definition 5.1. *A* commitment scheme *on ℓ bits is an efficient algorithm $\mathsf{Com} : \{0,1\}^\ell \times \{0,1\}^* \to \{0,1\}^*$ that takes as input (1) a plaintext $m \in \{0,1\}^\ell$; and (2) random coins $r \in \{0,1\}^*$ with the following two properties.*

- **(Perfectly binding.)** *For all $r, r' \in \{0,1\}^*$ and any $m, m' \in \{0,1\}^\ell$ with $m \neq m'$, $\mathsf{Com}(m;r) \neq \mathsf{Com}(m';r')$.*
- **(Computationally hiding.)** *For any PPT adversary $\mathcal{A}$, there exists a negligible function $\varepsilon(n)$ such that for all n and all $m_0, m_1 \in \{0,1\}^\ell$,*

$$\Pr_{b \sim \{0,1\}, r \sim \{0,1\}^n}[b' \leftarrow \mathcal{A}(1^n, \mathsf{Com}(m_b; r)),\ b' = b] \leq 1/2 + \varepsilon(n) \ .$$

Theorem 5.2. *If Com is a perfectly binding and computationally hiding commitment scheme on one bit, then for any ℓ, the function Com^ℓ defined by*

$$\mathsf{Com}^\ell((m_1, \ldots, m_\ell); (r_1, \ldots, r_\ell)) = (\mathsf{Com}(m_1; r_1), \ldots, \mathsf{Com}(m_\ell; r_\ell))$$

is a perfectly binding and computationally hiding commitment scheme on ℓ bits.

Proof. It is immediate that Com^ℓ is perfectly binding, so we only need to show that it is computationally hiding.

[3] This kind of argument, in which we say that "when we call $\mathcal{A}$ on input (X, Y, Z), from her perspective it is the same as if we had called her on input (X', Y', Z')" is extremely common. In other words, we often show two different ways to sample from the same distribution, one that corresponds exactly to some security game, and another that we use in our reduction. We then use the fact that $\mathcal{A}$ must behave the same in both cases because the distributions are in fact the same.

As always, we assume for contradiction that it is not, i.e., that there exists some PPT adversary $\mathcal{A}$ and pair of messages $m_0 = (m_{0,1}, \ldots, m_{0,\ell}), m_1 = (m_{1,1}, \ldots, m_{1,\ell}) \in \{0,1\}^\ell$ such that

$$\varepsilon(n) := \Pr_{b\sim\{0,1\}, r\sim\{0,1\}^{\ell n}}[b' \leftarrow \mathcal{A}(1^n, \mathsf{Com}^\ell(m_b; r)),\ b' = b] - 1/2\ .$$

is non-negligible.

We wish to show that this implies the existence of an adversary $\mathcal{A}'$ that breaks the hiding property of Com. It is not immediately clear how to do this. (Think about it!)

The trick is to use a sequence of *hybrid* games. In particular, let $m^{(i)} := (m_{0,1}, \ldots, m_{0,i}, m_{1,i+1}, \ldots, m_{1,\ell})$ be the message whose first i bits are the first i bits of m_0 and last $\ell - i$ bits are those of m_1. In particular, $m^{(0)} = m_1$ and $m^{(\ell)} = m_0$.

By assumption, we know that

$$2\varepsilon(n) = \Pr[\mathcal{A}(1^n, \mathsf{Com}^\ell(m^{(0)})) = 1] - \Pr[\mathcal{A}(1^n, \mathsf{Com}^\ell(m^{(\ell)})) = 1]$$

is non-negligible. We claim that there exists an $i \in \{1, \ldots, \ell\}$ such that

$$\varepsilon_i := \Pr[\mathcal{A}(1^n, \mathsf{Com}^\ell(m^{(i-1)})) = 1] - \Pr[\mathcal{A}(1^n, \mathsf{Com}^\ell(m^{(i)})) = 1] \geq 2\varepsilon(n)/\ell\ ,$$

which is non-negligible. Indeed,

$$2\varepsilon(n) = \sum_i \varepsilon_i \leq \ell \max_i \varepsilon_i\ ,$$

which gives the result. We assume without loss of generality that $m_{0,i} = 1$ and $m_{1,i} = 0$.

Now, we build $\mathcal{A}'$ that behaves as follows. On input $c^* := \mathsf{Com}(b)$ for $b \sim \{0,1\}$, $\mathcal{A}'$ samples $c_1 \leftarrow \mathsf{Com}(m_{0,1}), \ldots, c_{i-1} \leftarrow \mathsf{Com}(m_{0,i-1}), c_i := c^*, c_{i+1} \leftarrow \mathsf{Com}(m_{1,i+1}, \ldots, c_\ell \leftarrow \mathsf{Com}(m_{1,\ell})$. I.e., $c_1, \ldots, c_{i-1}$ are commitments to the first $i-1$ bits of m_0, $c_{i+1}, \ldots, c_\ell$ are fresh independent commitments to the last bits of m_1, and $c_i = c^*$. $\mathcal{A}'$ then runs

$$b' \leftarrow \mathcal{A}(1^n, c_1, \ldots, c_\ell)$$

and outputs b'.

Clearly $\mathcal{A}'$ runs in polynomial time. Furthermore, notice that when $b = 0$, the input to $\mathcal{A}$ is distributed exactly as a fresh commitment to $m^{(i)}$, while when $b = 1$, the input to $\mathcal{A}$ is distributed exactly as a fresh commitment to $m^{(i-1)}$. Therefore,

$$\Pr[b' = b] = \frac{1}{2} + \frac{1}{2}\cdot\Pr[\mathcal{A}(1^n, \mathsf{Com}^\ell(m^{(i-1)}) = 1] - \frac{1}{2}\Pr[\mathcal{A}(1^n, \mathsf{Com}^\ell(m^{(i)}) = 1] = \frac{1}{2} + \varepsilon_i \geq \frac{1}{2} + \varepsilon(n)/\ell\ ,$$

which is non-negligible by assumption. This contradicts the assumption that Com is computationally hiding, and the result follows. □

References

[Blu83] Manuel Blum. Coin flipping by telephone—a protocol for solving impossible problems. *ACM SIGACT News*, 15(1), 1983. 1, 4

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4), 1999. 1

[SRA79] Adi Shamir, Ron Rivest, and Leonard Adleman. Mental poker. MIT technical report, 1979. 4