0368-4170-01 Crypography and Game Theory Ran Canetti and Alon Rosen

Lecture 2

28 October 2009 Fall 2009

Scribe: Shai Vardi

Today

- Computational notions of security
- Pseudo-random generators
- One-way functions
- Commitment protocols

1 Security

Suppose Alice sends an encrypted message to Bob $c = E_k(m)$, where m is the plaintext message, c is the ciphertext and E is the encryption with respect to key k. If an eavesdropper Eve intercepts the message, knowing the encryption and decryption schemes but not the key, how much can Eve learn about m? We formalize this using two measures of security: *indistinguishability of encryptions* and *semantic security*.

1.1 Security by Indistinguishability

Definition 1 An encryption scheme (G, E, D) has *indistinguishable encryptions* if for all (non-uniform) PPT A there exists a negligible function ϵ such that for all $m_0, m_1 \in P_n$ (where P_n is the plaintext space indexed by n)

$$\mid Pr[A(E_k(m_0)) = 1] - Pr[A(E_k(m_1)) = 1] \mid < \epsilon(n),$$

where the probability is taken over all $k \stackrel{R}{\leftarrow} G(1^n)$ and the coin tosses of A.

Informally, this means that as long as an adversary runs in polynomial time, he cannot distinguish non-negligibly between any two plaintext messages m_0 and m_1.

Assuming that super-polynomial time is infeasible, the definition of indistinguishable encryptions seems to offers satisfactory security. Note here how cryptography differs from Game Theory: in Game Theory, computational time is not usually limited.

1.2 Semantic Security

We now introduce an alternative definition of security for encryption. We would like to say that an adversary does not learn anything from seeing the ciphertext. This is formalized by saying that whatever the adversary could have learned with the ciphertext she could have learned without it.

Definition 2 An encryption scheme (G, E, D) satisfies *semantic security* if for all (non-uniform) PPT A there exists an A' such that for every distribtion M on P_n and every function $f : P_n \to \{0,1\}^*$, there exists a negligible function ϵ such that

$$Pr[A(1^n, E_k(M)) = f(M)] \leq Pr[A'(1^n) = f(M)] + \epsilon(n)$$

Again the probability is taken over $k \stackrel{R}{\leftarrow} G(1^n)$ and the coin tosses of A. A' is sometimes called a *simulator*.

Note that f can be *any* function, and not necessarily one that is efficiently computable.

Theorem 1 An encryption scheme satisfies *indistinguishable encryption* if and only if it satisfies *semantic security* (no proof given).

Definitions 1 and 2 exemplify two general approaches to capture the security of a cryptographic scheme. The first one is *indistinguishability* and the second one is *simulation*.
Theorem 1 gives us an indication of the robustness of the two definitions.

In the follwing sectoion we introduce a tool that enables us to take advantage of the 'relaxation' offered by indistinguishable encryption and semantic security, and to bypass the limitations of perfect security in terms of key length vs. message length.

2 Pseudorandom Generators

We recall that for perfect security we need a (perfectly random) key of length $\geq$ the length of the plaintext message. Because this is impractical, we relaxed the definition of perfect security into one of computational security. We will take a seed of length n and use it to make a key of length $\ell(n)$ that is not perfectly random, but rather *pseudo random*. Informally this means that the key is virtually indistinguishable from a perfectly random key, and thus will guarantee us computational security.

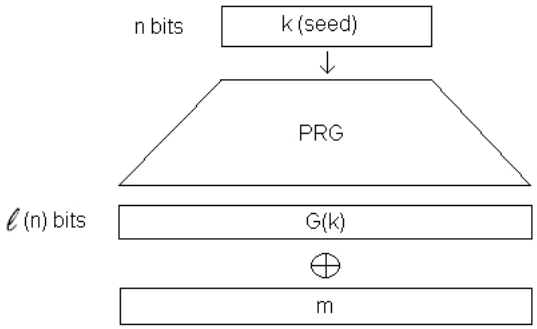

Figure 1: Pseudorandom Generator

Definition A function $G : \{0,1\}^* \rightarrow \{0,1\}^*$ is a *pseudorandom generator* (PRG) if

- *Efficiency*: G can be deterministically computed in polynomial time
- *Expansion*: $|G(x)| = \ell(|x|)$ for some expansion function satisfying $\ell(n) > n$
- *Pseudorandomness*: For any (non-uniform) PPT D (called a *distinguisher*) there exists a negligible function ϵ such that

$$| Pr[D(G(U_n)) = 1] - Pr[D(U_{\ell(n)}) = 1] | < \epsilon(n)$$

where U_n is the uniform distribution over n bits

Notice that the output of a PRG is statistically far from the uniform distribution on $\ell(n)$ bits.

2.1 Using Pseudorandom Generators

We will use a PRG G to encode a message with indistinguishable encryptions:

The encryption scheme is for messages of length $\ell(n)$ and operates as following:

- *Key Generation* : $G_{enc}(1^n)$ picks a random n-bit seed k for G
- *Encryption* : $E_k(m) = m \oplus G(k)$
- *Decryption* : $D_k(c) = c \oplus G(k)$

Theorem If G is a PRG then (G_{enc}, E, D) has indistinguishable encryptions against a (non uniform) PPT adversary.

Proof Let A be any (non-uniform) PPT algorithm and let m_0 , $m_1 \in P_n = \{0,1\}^{\ell(n)}$. We want to show that the encryptions of m_0 and m_1 are computationally indistinguishable.

We define 4 distributions:

1. $\mathbf{Real_0} \equiv \mathbf{E_k(m_0)} = \mathbf{G(k)} \oplus \mathbf{m_0} \quad (k \xleftarrow{R} U_n)$
2. $\mathbf{Ideal_0} \equiv \mathbf{k'(m_0)} = \mathbf{k'} \oplus \mathbf{m_0} \quad (k' \xleftarrow{R} U_{l(n)})$
3. $\mathbf{Ideal_1} \equiv \mathbf{k'(m_1)} = \mathbf{k'} \oplus \mathbf{m_1} \quad (k' \xleftarrow{R} U_{l(n)})$
4. $\mathbf{Real_1} \equiv \mathbf{E_k(m_1)} = \mathbf{G(k)} \oplus \mathbf{m_1} \quad (k \xleftarrow{R} U_n)$

We want to show that $\mathbf{Real_0}$ and $\mathbf{Real_1}$ are computationally indistinguishable. We do this as follows:

1 $\mathbf{Real_0}$ and $\mathbf{Ideal_0}$ are computationally indistinguishable by the pseudorandomness of G (see proof of claim below).

2 $\mathbf{Ideal_0}$ and $\mathbf{Ideal_1}$ are identically distributed by perfect secrecy of OTP.

3 $\mathbf{Real_1}$ and $\mathbf{Ideal_1}$ are computationally indistinguishable by the pseudorandomness of G (proved in an analogous way to claim 1).

By the triangle inequality, we have:

$$\begin{aligned}
&| Pr_k[A(\mathbf{Real_0}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Real_1}) = \mathbf{1}] | \leq \\
&| Pr_k[A(\mathbf{Real_0}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Ideal_0}) = \mathbf{1}] | + \\
&| Pr_k[A(\mathbf{Ideal_0}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Ideal_1}) = \mathbf{1}] | + \\
&| Pr_k[A(\mathbf{Ideal_1}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Real_1}) = \mathbf{1}] |
\end{aligned}$$

By 1) and 3), we have that $| Pr_k[A(\mathbf{Real_0}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Ideal_0}) = \mathbf{1}] |$ and $| Pr_k[A(\mathbf{Ideal_1}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Real_1}) = \mathbf{1}] |$ are negligible because of the pseudorandomess of G, and 2), we have that $| Pr_k[A(\mathbf{Ideal_0}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Ideal_1}) = \mathbf{1}] |$ is negligible because of perfect secrecy.
Thus, we conclude that $| Pr_k[A(\mathbf{Real_0}) = \mathbf{1}] - \mathbf{Pr_k}[\mathbf{A}(\mathbf{Real_1}) = \mathbf{1}] |$ is negligible.

Claim $\mid Pr[A(\mathbf{Real_0}) = \mathbf{1}] - \mathbf{Pr}[\mathbf{A}(\mathbf{Ideal_0}) = \mathbf{1}] \mid < \mathbf{neg(n)}$

Proof Using the pseudorandomness of G, define $A'_{m_0} = A(y \oplus m_0)$.

Then,

$$Pr[A(\mathbf{Real_0}) = \mathbf{1}] = \mathbf{Pr}[\mathbf{A'_{m_0}}(\mathbf{G}(\mathbf{U_n})) = \mathbf{1}]$$
$$Pr[A(\mathbf{Ideal_0}) = \mathbf{1}] = \mathbf{Pr}[\mathbf{A'_{m_0}}(\mathbf{G}(\mathbf{U_{\ell(n)}})) = \mathbf{1}]$$

and so $\mid Pr[A'_{m_0}(G(U_n)) - Pr[A'_{m_0}(G(U_{\ell(n)})) = 1] \mid < neg(n)$ because A'_{m_0} runs in polynomial time and G is a PRG. Remark: The requirement of G's security against non-uniform adversaries is a result of our need to 'hardwire' $m_0 \in P_n$ for each n.

We can further conclude that computational indestiguishability is *transitive*.

2.2 Do pseudorandom generators exist?

PRGs may not seem likely to exist, as the imply the ability to create a key of length $\ell(n)$ from a key of length only n which is indistinguishable (under the assumptions given above) from a random key of length $\ell(n)$.

Nevertheless, some supporting evidence for the existence of PRGs is that they can be constructed from one-way functions, which, although not known to exist either, are a more credible assumption. We will not show this construction, but given that it exists, and as we have already shown $PRG \rightarrow ENC$, we have the follwing sequence of deductions:

$$OWF \rightarrow PRG \rightarrow ENC$$

This suggests that efficient and secure encryption may indeed exist.

3 One Way Functions

3.1 Definitons

Loosely speaking, a *one-way function (OWF)* is a function that is easy to compute but hard to invert, i.e. computing $x \longmapsto f(x)$ is easy while computing $f(x) \longmapsto f^{-1}(f(x))$ is hard.

One reason to believe that one-way functions exist is that they seem to exist in the physical realm, for example lighting a match and breaking glass are two actions (functions) that are easy to perform, but seem hard to reverse.

Definition $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is a *one-way function* if

- f can be evaluated in polynomial time
- For every PPT A there exists a negligible function ϵ such that

$$Pr[A(f(x), 1^n) \in f^{-1}(f(x))] < \epsilon(n)$$

The probability is over the choice of a random x and the coin tosses of A.

We need to define what we mean by 'inverting a function' in this context: the adversary A is considered to have successfully inverted function f if:

- A finds *some* pre-image of f
- A finds f^{-1} with probability $\frac{1}{poly(n)}$
- A finds f^{-1} for infinitely many n's

We are 'liberal' with what constitutes the success of an inverter, mostly because if we are not, there are many choices of f which would qualify as one-way functions by definition, but would not capture the 'essence' of one-way functions . We can afford to be liberal because if we succeed to satisfy one-wayness while being liberal about A's success then we certainly don't get a weaker function.

Some Observations :

- $\|f(x)\| \leq poly(\|x\|)$ This is because of the requirement that $f(x)$ can be computed in polynomial time.
- If we were to require that $A(f(x)) = x$, this would be too demanding. For example, if $f(x) = \|x\|$ then $Pr[A(f(x) = x] \leq \frac{1}{2^n}$. This is true because there are $O(2^n)$ values of x for which $\|x\|$ is identical.
- $Pr[A(f(x)) \in f^{-1}(f(x))]$ is not enough. We need the security parameter 1^n. Otherwise, if we (once again) use the example of $f(x) = \|x\|$, we have $\|f(x)\| = log(\|x\|)$ and to even write down $f^{-1}(f(x))$, A would need time exponential in his input length. (There are $O(n)$ values of x for which $f(x)$ is identical, and, as $\|f(x)\| = log\|x\|$, any inverting function g would need exponential time to invert it.) We obviously do not want this kind of function to qualify as a OWF. This 'unconditionally secure' OWF would not be very useful.

3.2 Examples of Candidate One-Way Functions

3.2.1 Multiplication

The following function is widely belived to be hard to invert.

$$f(x, y) = x * y, \text{ where } \|x\| = \|y\|$$

Assumption : For any PPT A there exists a negligible function ϵ such that $Pr[A(N) \in \{P, Q\}] \leq \epsilon(n)$, where P and Q are n-bit primes and $N = PQ$

We don't know if this assumption is true (this is usually referred to as the *factoring assumption*), but if it is, then it allows for secure encryption. Currently, the best provable asymptotic algorithm for factoring runs in time $exp(O(\sqrt{n}\sqrt{logn}))$.

3.2.2 Subset Sum

$$f(x_1, x_2, ..., x_n, S) = (x_1, ..., x_n, \textstyle\sum_{i \in S} x_i \bmod 2^n)$$

where each x_i is a random n-bit number and S is a random subset of $\{1...n\}$.

Subset Sum is known to be NP-complete. But this does not provide sufficient indication about its one-wayness, since NP-completeness is a worst-case notion, whereas one-wayness is an average-case notion. A nice property of the subset-sum function is that for any $\ell(n) > n$ for which it is one-way $\bmod 2^{\ell(n)}$, it is also a pseudorandom generator. [Impagliazzo, Naor '96].

4 Commitment Protocols

4.1 Motivation

Say Alice wants to prove to Bob that she can do something, like predict the result of a coin flip, but she doesn't want him to have the result beforehand, how can she do this? She does it by a 2-part process called a *commitment scheme*

1. *Commitment Phase* - Alice takes a prediction, puts it in a box and gives the locked box to Bob.
2. *Opening phase* - Alice gives Bob the key to the box.

For a commitment protocol to work, three properties must be satisfied:

1. *Completeness* - Bob will open the box and find the value that Alice committed to.
2. *Secrecy* - As long as Alice doesn't give the key to Bob, Bob cannot know what the prediction is.
3. *Binding* - Bob knows that as soon as he has the box, the prediction is fixed.

We will now formalize this intuitive concept. To do this, we will first need a basic model for representing protocols for two parties. (Later, we will extend this model to deal with more parties.) Such a model should capture the behavior of protocols in the presence of computationally bounded, adversarial participants. The model we'll use is Interactive Turing Machines.

4.2 Interactive Turing Machines

An *interactive turing machine (ITM)* is a turing machine with incoming and outgoing communication tapes.

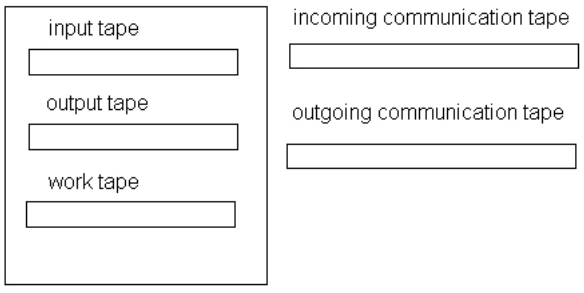

Figure 2: Interactive Turing Machine

If an ITM is probablistic (such as the ones we will be dealing with), it will also have a random tape. An ITM, as its name suggests, interacts with other ITMs, via the input and output communications tapes. Each ITM works thus: ITM A recieves an input on the 'input' tape, like a standard turing machine. It then performs computations like a standard turing machine, in which it can write on any of its tapes, including the output communication tape, and at some stage it moves into a state 'send'. Then it 'sends' the data on the output communication tape and waits for input on the input communication tape. When the ITM it is interacting with goes into the 'send' state, A recieves the data on the input communication tape and continues computing. Obviously, usually an ITM will not

work alone - there will be 2 (or more) ITMs working together (see figure 3), where one ITM's input communication tape is another ITM's output communication tape and vice versa. Then ITM B starts in a waiting state and A perfoms a computation, moves into the state 'send' and sends some data to ITM B. B then performs some computation depending on its input and the input it just received from A, sends data to A and waits again. This is repeated until they both finish computing. Note also that it is possible to perform computations in several stages, where after writing the output, the internal state of both ITMs does not change, and they wait for more external inputs, and then continue computing once they receive the input. This will be useful for modeling commitment protocols, where, as we saw, there is more than one phase.
An ITM is considered polynomial if it is polynomial in the input (the standard input, not the input communication tape).

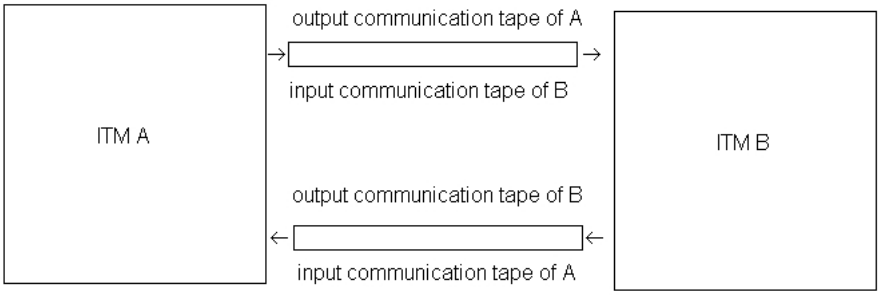

Figure 3: Interaction between two Interactive Turing Machines

Notation $[A, B](x_A^1, x_B^1; x_A^2, x_B^2; ...; x_A^n, x_B^n) = (y_A^1, y_B^1; ...; y_A^n, y_B^n)$

Here A and B are two ITMs, x_T^i is the input to ITM T in phase i of the communication and y_T^i is is the output of ITM T in phase i of the communication. (Note that *input* and *output* refer to external input and output, not to the communication tapes.) Note that if A and B are probablistic, then the y's are random variables.

4.3 Commitment Scheme

The I/O interface for a commitment scheme is as follows: (C and R are the *committer* and *receiver* respectively)

Phase 1: (commitment phase)

$x_C^1 = m(prediction), 1^n$	$y_C^1 = \perp$
$x_R^1 = \perp$	y_R^1 = 'committed'

Phase 2: (opening phase)

x_C^2 = 'open'	$y_C^2 = \perp$
$x_R^2 = \perp$	y_R^2 = m' / error

Where m' should (hopefully) equal m, and 1^n is added to ensure polynomiality. If we did not require 1^n in the input, we could, for example, have $m \in \{0, 1\}$ and then any computation we did would be at least exponential in the input length (for example, in the commitment scheme we will shortly

describe, C computes a function $\{0,1\}^n \to \{0,1\}^{3n}$. If we will not add 1^n to the input, C will not be able to compute this function in polynomial time).
In words, the commitment scheme proceeds as follows: In phase 1 (commitment), the committer (C) receives a value (m), which it needs to commit to. Then some computation can happen (including interaction between C and R - the receiver). Then when the computation and interaction is over for phase 1, R outputs 'committed'. The beginning of phase 2 (opening) is marked by C receiving the input 'open' and then some computing occurs (which will usually include sending R the key and R opening the encryption). At the end of this phase (and the protocol), R will output the message it received, m', (hopefully $m' = m$) or 'error' if it could not correctly open the committed value.

Definiton A pair (C, R) is a *commitment scheme* with *computational security* if the following three conditions hold:

- *Completeness* : $[C, R]((1^n, m), (1^n, \perp); \text{`}open', \perp) = (\perp, \text{`}committed'; \perp, m)$
- *Secrecy* : For any PPT R^*, there exists a negligible function ϵ such that

$$Pr(b \leftarrow \{0,1\} : [C, R^*]((1^n, b), (1^n, \perp)) = (\perp, b)) < \tfrac{1}{2} + \epsilon(n)$$

- *Binding* : For any PPT C^*, there exists a negligible function ϵ such that

$$Pr(b \leftarrow \{0,1\} : [C^*, R](1^n, 1^n; b, \perp) = (\perp, \text{`}committed'; \perp, b)) < \tfrac{1}{2} + \epsilon(n)$$

In this definition we only consider commitment scheme s, where Alice commits to a single bit. This definition can easily be modified to accomodate for an arbitrary number of bits.
Let us briefly explain why these definitions for *Completeness*, *Secrecy* and *Binding* capture the intuitive notions described in 4.1. *Completeness* follows directly from the definition of the commitment scheme - if the protocol is followed by both parties, at the end of the protocol, R ouputs (knows) M. In the definition of *Secrecy*, we are basically saying, that without C saying 'open', the chances of any R^* to find b are small. [For both *Secrecy* and *Binding*, we use $< \frac{1}{2} + \epsilon(n)$ because the chances of guessing b are $\frac{1}{2}$ in any case, and in fact, if the odds are $= \frac{1}{2}$, we will have perfect secrecy (more on that in the next paragraph).]

We would like to require that at the end of the commitment phase there will be a unique value u, such that for every $u' \neq u$, the probability of R opening u' is negligible. However, this requirement is difficult to formalize, as it is not always clear what the value of u is. (Specifically, the value of u can depend on some secret infiormation that C holds, and not only on the communication.) Therefore, we will require something slightly different: we will describe a game, in which after the receiver outputs 'committed', the committer C^* receives a challenge v. C^* wants to make R open the value v, but succeeds only with probability $\frac{1}{2}$ (over the choice of v). This guarantees that at the end of the commitment phase, there is a single value that C^* can successfully cause R to open. Indeed, if there was more than one such value, C^* could win the game with probability $> \frac{1}{2}$.

In the definition of *Binding*, C^* receives b *after* R outputs 'committed', meaning it recived it after the commitment stage. Now it has to get R to open this b. To ensure binding, it should be hard for C^* to do this.

Observations

In a perfect commitment scheme, there should be only one value R could open. In that case, we would have, for any R^*

$$Pr(b \leftarrow \{0,1\} : [C, R^*](1^n, b; 1^n, \perp) = (\perp, b)) = \tfrac{1}{2},$$

giving us perfect secrecy, and for any $C*$,

$$Pr(b \leftarrow \{0,1\} : [C^*, R](1^n, 1^n; b, \perp) = (\perp, `committed'; \perp, b)) = \tfrac{1}{2},$$

giving us perfect binding. However, so we need to define the 'commitment game' to allow for statistical and computational security. Statistical security is the same as computational security, but for *any* R^* (not just PPT) for *statistical secrecy* and for any C^* for *statistical binding*.

Theorem There is no commitment scheme which has both statistical secrecy and statitistical binding (no proof given).

4.4 Commitment based on PRG (Naor '91)

Alice wants to commit to a bit b to Bob. We will describe a commitment scheme using G, a pseudorandom generator.

Protocol

let $G : \{0,1\}^n \to \{0,1\}^{3n}$ be a PRG.

choose $r \in \{0,1\}^n$	C	$\xleftarrow{s}$	R $s \xleftarrow{R} \{0,1\}^{3n}$
		if $b = 0$, $t = G(r)$ if $b = 1$, $t = G(r) \oplus s$	stores s & t
"commit"		$\longrightarrow$	'committed'
"open"		$\xrightarrow{r}$	if $t = G(r)$, R accepts 0 if $t = G(r) \oplus s$, R accepts 1

In words, the protocol is as follows: C receives a bit, b, which it wants to commit to R. R uniformly chooses s from $\{0,1\}^{3n}$ and sends s to C. C then uniformly chooses a seed, r, and generates a random key, $G(r)$, using G. In the *commitment phase* - depending on whether $b = 0$ or $b = 1$, C sends $G(r)$ or $G(r) \oplus s$ to R. R stores the message from C (which we call t), and outputs 'committed'. In the *opening phase*, C sends r to R, and R calculates $G(r)$ and accepts 0 or 1 depending on whether $t = G(r)$ or $t = G(r) \oplus s$.

Theorem $[C, R]$ is a commitment scheme with computational secrecy and statistical binding.

To show that this protocol is indeed a commitment scheme, we must prove completeness, secrecy and binding.

Proof

- Completeness : Trivial - if both C and R follow the protocol, then R can indeed learn b from the protocol.
- Secrecy : Follows from the fact that G is a PRG. Informally: For every $s \in \{0,1\}^{3n}$, U_{3n} and $U_{3n} \oplus s$ are identically distributed. Thus, if it is feasible for R^* to find an s such that $G(U_n)$ and $G(U_n) \oplus s$ are computationally distinguishable, then either $G(U_n)$ and U_{3n} or $G(U_n) \oplus s$ and $U_{3n} \oplus s$ are computationally distinguishable (or both), in contradiction to the pseudorandomness of G. More formally:

  Assume that there exists an R' such that for all negligible functions ϵ,

$$Pr(b \leftarrow \{0,1\} : [C, R'](1^n, b; 1^n, \perp) = (\perp, b)) \geq \frac{1}{2} + \epsilon(n)$$

Then, with probability $\geq \epsilon(n)$, R' distinguishes between $G(r)$ and $G(r) \oplus s$ (for an s that R' comes up with); that is, R' can tell with non-negligible probabililty, for any x, if $x \oplus s$ is in the range of G: We will use R' to build a distinguisher for G:

Reduction We will use R' to build a distinguisher D for G as follows: D receives an input y. R' randomly chooses $s \in \{0,1\}^{3n}$ and calculates $y \oplus s$. If R' accepts 1, then D concludes that $y \in range(G(r))$.

Proof of correctness A distinguisher D receives y and decides with non-negligible probability if $y \in G(r)$. We will build a distinguisher D as follows: we take R' from before, randomly select s from $\{0,1\}^{3n}$ and input $x = y \oplus s$ to R'. If R' accepts 1, then $y \in G(r)$. As R' can tell with non-negligible probability if $x \in G(r)$, then D can distinguish between Gr and U_{3n} with non-negligible probability, in contradiction to the pseudorandomness of G.

- Binding : Looking at the definition of *binding*, we see that any C' that receives b after R outputs 'commited' should not be able to convince R that the message he committed to was b with probability $\geq \frac{1}{2} + \epsilon(n)$, for any non-negligible ϵ(n). Assume such a C' exists. For C' to be able to convince R with probability $\geq \frac{1}{2} + \epsilon(n)$, he must be able to choose seeds r_1 and r_2 such that $G(r_1) = G(r_2) \oplus s$ with probability $\geq \epsilon(n)$ for the s chosen by R (4).
  Note that if C' does not choose such r's, there is nothing he can send to R in the opening phase that will enable R to correctly get either 1 or 0.
  We will now show that, because R chooses an s uniformly from $\{0,1\}^{3n}$, C' will almost never be able to choose such seeds r_1 and r_2.

  Definition A value $s \in \{0,1\}^{3n}$ if *bad* if there exist two seeds, $r_1, r_2 \in \{0,1\}$, such that $s \neq G(r_1) \oplus G(r_2)$

  For any pair of seeds, there is a unique s such that $s = G(r_1) \oplus G(r_2)$.
  There are 2^n distinct possibilites for r_1 and 2^n distict possibilites for r_2, therefore there are 2^{2n} pairs of seeds that C' can choose. Therefore there are at most 2^{2n} bad s's, and, as s's sample space is 2^{3n}, R has only a $\frac{1}{2^n}$ chance of choosing a bad s. This means, that even if C' knows how to choose r_1 and r_2 once given s, he will be able to choose r_1 and r_2 with probabilty $\leq \frac{1}{2^n}$. Taking ϵ as $\epsilon(n) = \frac{2}{2^n}$, we get a contradiction to (4). □