CMSC 858K — Advanced Topics in Cryptography	April 20, 2004
Lecture 22	
Lecturer: Jonathan Katz	*Scribe(s): Nagaraj Anthapadmanabhan, Ji Sun Shin*

1 Introduction to These Notes

In the previous lectures, we saw the construction of a zero-knowledge proof system for the language of 3-colorable graphs (and hence for all of NP) using a *commitment scheme* as a tool. In this lecture we will discuss the two different notions of commitment schemes and show how these types of commitment schemes can be constructed.

Recall that a commitment scheme consists of two phases: a *commitment phase* after which the sender is supposed to be committed to a value, and a *decommitment phase* during which the committed value is revealed to the receiver. Two types of commitment schemes can be defined, one protecting against an all-powerful *sender* and one protecting against an all-powerful *receiver*:

- A **standard commitment scheme** protects against a PPT receiver and an all-powerful sender. In particular, a PPT receiver cannot determine any information about the value committed to by the sender (this can be defined in a way similar to indistinguishability for encryption) in the commitment phase, and an all-powerful sender is bound to only one value (with all but negligible probability) following the commitment phase. We say such a scheme is *computationally hiding* and *information-theoretically binding*.
- A **perfect commitment scheme** protects against a PPT sender and an all-powerful receiver. Namely, even an all-powerful receiver cannot determine any information about the value committed to by the sender (except possibly with negligible probability) in the commitment phase, and a PPT sender is bound to only one value following the commitment phase. We say it is *computationally binding* and *information-theoretically hiding*.

It can be proved that commitment schemes which are both information-theoretically hiding and information-theoretically binding do not exist.

2 Standard Commitment Schemes

We first consider some constructions of standard commitment schemes.

2.1 Constructions Based on One-Way (Trapdoor) Permutations

One possible construction is obtained by using a *trapdoor permutation family*. Here we show how this can be used to allow a sender $\mathcal{S}$ to commit to a bit b:

$$
\begin{array}{lcl}
\underline{\mathcal{S}(1^k, b)} & & \underline{\mathcal{R}(1^k)} \\
\text{choose coins } \omega \text{ for } \mathsf{Gen} & & \\
(f, f^{-1}) \leftarrow \mathsf{Gen}(1^k; \omega) & & \\
r \leftarrow \{0,1\}^k & \xrightarrow{f,\ c = f(r),\ d = h(r) \oplus b} & \\
\hline
& \xrightarrow{f^{-1},\ r,\ \omega} & \text{check } (f, f^{-1}) \stackrel{?}{=} \mathsf{Gen}(1^k; \omega); \\
& & \text{check } f(r) \stackrel{?}{=} c; \\
& & b = d \oplus h(r)
\end{array}
$$

To commit to a bit b, the sender $\mathcal{S}$ generates a trapdoor permutation (f, f^{-1}), chooses a random element r in the domain of f (here, the domain of f is assumed to be $\{0,1\}^k$), and sends a description of f along with $f(r)$ and $h(r) \oplus b$ (here, $h(\cdot)$ is a hard-core bit of f). To decommit, $\mathcal{S}$ sends the random string ω (used to prove that f is indeed a permutation), as well as r. The receiver $\mathcal{R}$ does the verification in the obvious manner and recovers b.

The *correctness* property is obviously satisfied as the receiver $\mathcal{R}$ "accepts" if $\mathcal{S}$ acts honestly. (This will be the case for all constructions in this lecture.) We now sketch why the other two properties hold:

Binding: We need to prove binding even for an all-powerful sender. Notice that the f sent in the first round must be a permutation, since the sender has to reveal the randomness ω used to generate f in the decommitment phase (and we assume that Gen always outputs a permutation). Thus, c uniquely determines a value $r = f^{-1}(c)$, and this in turn uniquely determines $b = h(r) \oplus d$.

Hiding: Since f is one-way, a PPT receiver cannot distinguish $h(r)$ from a random bit (we use here the fact that (f, f^{-1}) are randomly-generated, and that r is chosen uniformly at random). Thus, $h(r)$ computationally hides the value of b. A formal proof is left as an exercise for the reader.

The reader may notice that we never use the trapdoor f^{-1} in the above construction. Thus, we may in fact base the above construction on the (potentially) weaker assumption of a one-way (*not* necessarily trapdoor) permutation.

2.2 A Construction Based on a One-Way Function

Actually, we can do even better and show a construction based on the minimal assumption of a one-way function (the proof that this assumption is indeed minimal is left as another exercise!). Here, we use the following "hard" theorem that was mentioned, but not proved, in class:

Theorem 1 *The existence of a one-way function implies the existence of a length-tripling pseudorandom generator (PRG).*

Recall the definition of a (length-tripling) PRG:

Definition 1 $G = \{G_k : \{0,1\}^k \to \{0,1\}^{3k}\}_{k\in\mathbb{N}}$ is a *pseudorandom generator (PRG)* if the following is negligible for any PPT distinguisher D:

$$\left|\Pr\left[x \leftarrow \{0,1\}^{3k} : D(x) = 1\right] - \Pr\left[s \leftarrow \{0,1\}^k; x = G_k(s) : D(x) = 1\right]\right|. \tag{1}$$

In other words,

$$U_{3k} \approx G(U_k) \tag{2}$$

where U_k denotes the uniform distribution on k-bit strings. ♢

We would like to use a PRG to construct a standard commitment scheme. We will first see a naïve idea that does not work.

$\underline{\mathcal{S}(1^k, b)}$		$\underline{\mathcal{R}(1^k)}$
if b=0 {		
$s \leftarrow \{0,1\}^k$		
$x = G(s)$}		
if b=1		
$x \leftarrow \{0,1\}^{3k}$	$\xrightarrow{x}$	
	– – – – – – – – – – – – – –	
if b=0		
decom $= s$		
if b=1		
decom = "random"	$\xrightarrow{decom}$	Check decommitment

I.e., $\mathcal{S}$ sends a pseudorandom string to commit to "0", and a random string to commit to "1". To decommit, $\mathcal{S}$ sends the seed if x was pseudorandom (i.e., $b = 0$) or just says that x is "random" otherwise (i.e., $b = 1$).

It is easy to see that the above scheme satisfies hiding: this follows readily from the definition of a PRG. Unfortunately, the scheme is not binding, even for a polynomial-time sender. This is because a cheating $\mathcal{S}^*$ can send a pseudorandom string $x = G(s)$ and later decommit this to either a 0 (by sending s) or a 1 (by claiming that x is a random string). Note that the properties of the PRG imply that a PPT $\mathcal{R}$ cannot even tell that $\mathcal{S}^*$ is cheating!

Informally, the problem arises from the fact that the sender is not required to use any "secret" when committing to $b = 1$. So, we fix this by making the sender use a secret for either value of b. The following construction is due to Naor [1]:

$\underline{\mathcal{S}(1^k, b)}$		$\underline{\mathcal{R}(1^k)}$
$s \leftarrow \{0,1\}^k$	$\xleftarrow{\quad r_0 \quad}$	$r_0 \leftarrow \{0,1\}^{3k}$
$x = G(s)$		
if b=0		
$\quad y = x = G(s)$		
if b=1		
$\quad y = x \oplus r_0$	$\xrightarrow{\quad y \quad}$	
	$- - - - - - - - - - - - - - -$	
	$\xrightarrow{\quad s,\ b \quad}$	if b=0
		$\quad$ check $y \stackrel{?}{=} G(s)$
		if b=1
		$\quad$ check $y \stackrel{?}{=} G(s) \oplus r_0$

The receiver first sends a random $3k$-bit string r_0. The sender chooses a random seed s and computes $x = G(s)$. The sender simply sends x to commit to "0", and sends $x \oplus r_0$ to commit to "1". To decommit, $\mathcal{S}$ sends both s and the committed value (and the receiver verifies these in the obvious way). Clearly, this construction satisfies the correctness property. We claim that this construction also satisfies the hiding and binding requirements:

Hiding: The following claim proves the hiding property:

Claim 2 *If a* PPT *cheating receiver $\mathcal{R}^*$ can distinguish a commitment to 0 from a commitment to 1 (before the decommitment phase), then we can use $\mathcal{R}^*$ to "break" the PRG.*

Proof Let $\mathcal{R}^*$ be a PPT receiver that distinguishes a commitment to 0 from a commitment to 1 with probability $\varepsilon(k)$ (namely, the difference between the probability that $\mathcal{R}^*$ outputs 1 when interacting with a sender committing to a "0" and the probability that $\mathcal{R}^*$ outputs 1 when interacting with a sender committing to a "1" is $\varepsilon(k)$). We want to show that $\varepsilon(k)$ is negligible. To that end, use $\mathcal{R}^*$ to construct a PPT distinguisher D trying to distinguish whether its input x is random or pseudorandom:

$\underline{D(x)}$
$b \leftarrow \{0,1\}$
run the commitment phase of the above scheme with $\mathcal{R}^*$ using bit b; i.e.:
  get r_0 from $\mathcal{R}^*$
  if $b = 0$ send x to $\mathcal{R}^*$
  if $b = 1$ send $x \oplus r_0$ to $\mathcal{R}^*$
if $\mathcal{R}^*$ guesses b correctly, then D outputs 1 (i.e., "PRG")
otherwise, D outputs 0 (i.e., "random"

Note that when x is pseudorandom, D acts as a completely honest sender and so the probability that D outputs 1 is given by:

$$\frac{1}{2} \cdot (\Pr[\mathcal{R}^* \text{ outputs } 0 \mid \mathcal{S} \text{ commits to } 0] + \Pr[\mathcal{R}^* \text{ outputs } 1 \mid \mathcal{S} \text{ commits to } 1])$$

$$
\begin{aligned}
&= \frac{1}{2} \cdot (1 - \Pr[\mathcal{R}^* \text{ outputs } 1 \mid \mathcal{S} \text{ commits to } 0] + \Pr[\mathcal{R}^* \text{ outputs } 1 \mid \mathcal{S} \text{ commits to } 1]) \\
&= 1/2 + \varepsilon(k)/2.
\end{aligned}
$$

On the other hand, when x is random then the view of $\mathcal{R}^*$ is independent of b (since either one of x or $x \oplus r_0$, for any r_0, is just a uniformly-distributed string) and so the probability that D outputs 1 is exactly 1/2. The security of the PRG thus implies that $1/2 + \varepsilon(k)/2 - 1/2 = \varepsilon(k)/2$ is negligible, completing the proof. ■

Binding: Proving the binding property is more interesting. We show that an all-powerful $\mathcal{S}^*$ can decommit to two different values with only negligible probability (assuming $\mathcal{R}$ is honest). Note that $\mathcal{S}^*$ can only potentially cheat if $\mathcal{R}$ sends an r_0 for which there exist y, s, s' such that: $y = G(s)$ and $r_0 = y \oplus G(s')$ or, in other words, if there exist s, s' such that $G(s) \oplus G(s') = r_0$. Call r_0 with this property "bad". Because s and s' are k-bit strings, there are at most 2^{2k} possible "bad" values r_0. Since r_0 is a uniformly-random $3k$-bit string, the probability of $\mathcal{R}$ choosing a "bad" r_0 is at most $2^{2k}/2^{3k} = 2^{-k}$, which is negligible.

3 Perfect Commitment Schemes

Perfect commitment schemes seem much more difficult to construct. In particular, a construction of a perfect commitment scheme based on one-way *permutations* is known, but it is an open question to construct such a scheme based on one-way functions. Here, we show two constructions based on number-theoretic assumptions.

3.1 A Construction Based on the Discrete Logarithm Assumption

The following scheme allows a sender to commit to a string $x \in \{0,1\}^{k-1}$, not just a bit:

$\underline{\mathcal{S}(1^k, x)}$		$\underline{\mathcal{R}(1^k)}$
		generate a cyclic group G of prime order q with $\lvert q \rvert = k$
	$\xleftarrow{\quad g,\ h,\ G,\ q \quad}$	pick random generators $g, h \in G$
verify q prime; $\lvert q \rvert = k$ verify g, h generators of G view x as a value in $\mathbb{Z}_q$ $r \leftarrow \mathbb{Z}_q$		
$c = g^x h^r$	$\xrightarrow{\quad c \quad}$	
	– – – – – – – – – – – –	
	$\xrightarrow{\quad x,\ r \quad}$	verify $c \stackrel{?}{=} g^x h^r$

Note that in a cyclic group of prime order, every element except the identity is a generator.

We now show that the above constitutes a perfect commitment scheme:

Hiding: We claim that c is a uniformly-distributed group element, independent of the committed value x. To see this, note that the sender verifies that q is prime and that g, h are generators of G (also, $\mathcal{S}$ implicitly verifies that G is a cyclic group of prime order). Fixing x and considering any element $\mu \in G$, we calculate the probability that $c = \mu$ (where the probability is over the sender's choice of r). Let $\alpha = \log_g \mu$ and $\beta = \log_g h$ (these are well-defined, since g is a generator and G is cyclic). Then $c = \mu$ if and only if $\alpha = \log_g \mu = \log_g(g^x h^r) = x + r\beta \bmod q$ or, equivalently, if $r = (\alpha - x)\beta^{-1} \bmod q$ (note that $\beta^{-1} \bmod q$ is defined, since h is a generator so $\log_g h \neq 0$). But the probability that r takes on this value is exactly $1/|\mathbb{Z}_q| = 1/q = 1/|G|$. This implies that the value of x is perfectly hidden.

Another way to view this ("in reverse") is to note that for any value c there are exactly q possible pairs (x, r) satisfying $g^x h^r = c$, one for each possible value of $x \in \mathbb{Z}_q$. So, even an all-powerful $\mathcal{R}^*$ cannot tell which value of x is the one committed to by $\mathcal{S}$.

Binding: We show that if a PPT sender $\mathcal{S}^*$ can decommit to two different values, then we can use $\mathcal{S}^*$ to break the discrete logarithm assumption in G. On input G, q, g, h, algorithm A (with the goal of computing $\log_g h$) sends G, q, g, h to $\mathcal{S}^*$ who responds with a value $c \in G$. If $\mathcal{S}^*$ is now able to decommit to (x, r) *and* (x', r'), with $x \neq x'$ and $g^x h^r = c = g^{x'} h^{r'}$, then A can compute the desired discrete logarithm by noting that:

$$g^x h^r = g^{x'} h^{r'} \iff g^{x-x'} = h^{r'-r} \iff g^{(x-x')/(r'-r)} = h,$$

or $\log_g h = (x-x')(r'-r)^{-1} \bmod q$. (Note that $(r'-r)$ is non-zero as long as h is a generator — if not, then it is easy for A to compute $\log_g h$!). Clearly, A runs in polynomial time if $\mathcal{S}^*$ does; also, if $\mathcal{S}^*$ decommits to two different values with probability $\varepsilon(k)$ then A correctly computes $\log_g h$ with the same probability. Since this must be negligible under the discrete logarithm assumption, the binding property follows.

3.2 A Construction Based on the RSA Assumption

The next construction is based on the RSA assumption (for large prime public exponents), and again allows the sender to commit to a string $x \in \{0,1\}^k$.

$\underline{\mathcal{S}(1^k, x)}$		$\underline{\mathcal{R}(1^k)}$
		generate k-bit primes p, q set $N = pq$ choose a prime e s.t. $e > N$
verify e prime; $e > N$; $\mu \in \mathbb{Z}_N^*$	$\xleftarrow{\quad N,\ e,\ \mu \quad}$	$\mu \leftarrow \mathbb{Z}_N^*$
view x as an element of $\mathbb{Z}_e$ $r \leftarrow \mathbb{Z}_N^*$		
$c = \mu^x r^e \bmod N$	$\xrightarrow{\quad c \quad}$	
	– – – – – – – – – – – –	
	$\xrightarrow{\quad x,\ r \quad}$	verify $x \in \mathbb{Z}_e$ and $c \stackrel{?}{=} \mu^x r^e \bmod N$

We now prove both hiding and binding:

Hiding: Before sending the commitment, the sender verifies that e is prime and $e > N$; this guarantees that $\gcd(e, \phi(N)) = 1$. (We require $e > N$ because the receiver might try to cheat and send a modulus N which is not a product of two primes.) Thus, the function $f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ defined by $f(x) = x^e \bmod N$ is a permutation. Since r is chosen uniformly at random in $\mathbb{Z}_N^*$, the value $f(r) = r^e \bmod N$ is uniformly distributed in $\mathbb{Z}_N^*$. Since $\mu \in \mathbb{Z}_N^*$ and hence $\mu^x \in \mathbb{Z}_N^*$, the product $c = \mu^x r^e$ is uniformly distributed in $\mathbb{Z}_N^*$ and reveals no information about x.

Another way to understand this is that for any commitment $c \in \mathbb{Z}_N^*$ and for any possible $x \in \mathbb{Z}_e$, there exists an $r \in \mathbb{Z}_N^*$ such that $c = \mu^x r^e \bmod N$. So no information about x is revealed.

Binding: We show that if a cheating sender $\mathcal{S}^*$ can decommit to two different values, then we can use it to break the RSA assumption (for large public exponents). Namely, given (N, e, μ) with N a randomly-generated product of two primes, $e > N$ a prime, and μ chosen at random from $\mathbb{Z}_N^*$, we show how to use $\mathcal{S}^*$ to compute $\mu^{1/e}$. Simply send (N, e, μ) to $\mathcal{S}^*$ and assume it sends back $c \in \mathbb{Z}_N^*$ and can decommit to (x, r) and (x', r') such that $x \neq x'$, $\mu^x r^e = c = \mu^{x'}(r')^e \bmod N$, and both $x, x' \in \mathbb{Z}_e$. Without loss of generality assume $x > x'$. We know that:

$$\mu^x r^e = \mu^{x'}(r')^e \bmod N \iff \mu^{x-x'} = (r'/r)^e \bmod N.$$

Let $\Delta \stackrel{\text{def}}{=} x - x'$. Using the fact that $x, x' \in \mathbb{Z}_e$ and e is prime, we see that $\Delta < e$ and hence $\gcd(\Delta, e) = 1$. Using the extended Euclidean algorithm, we can compute in polynomial time integers A, B such that $A\Delta + Be = 1$ (over the integers). Then:

$$\begin{aligned}
\mu^1 = \mu^{A\Delta + Be} &= \left(\mu^{\Delta}\right)^A \mu^{Be} \bmod N \\
&= \left((r'/r)^e\right)^A (\mu^B)^e \bmod N \\
&= \left((r'/r)^A \mu^B\right)^e \bmod N,
\end{aligned}$$

and so

$$\mu^{1/e} = (r'/r)^A \mu^B \bmod N.$$

This algorithm runs in polynomial time if $\mathcal{S}^*$ does, and outputs the desired inverse with the same probability that $\mathcal{S}^*$ can decommit to two different values. Since the former probability is negligible under the RSA assumption (for large prime exponents), the binding property follows.

References

[1] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology* 4(2): 151–158 (1991).