Computer Science 308-647B
Advanced Cryptography

Claude Crépeau

These notes are, largely, transcriptions by the students of class notes from the course *Advanced Cryptography (308-647B)* that was given by prof. Claude Crépeau at McGill University during the winter of 2000.

1 Lecture 1

1.1 Interactive Proofs and Zero-Knowledge proofs

1.1.1 Interactive Proof (IP)

Protocols are given for allowing a prover P to convince a verifier V that P knows some verifiable secret information.
P wants to convince V that an element x is part of a language L , by interacting with V. Assume P is powerful (i.e. can work for as long as she wants) and V works in probabilistic polynomial time (relative to the input).

Consider the *GRAPH-ISOMORPHISM* problem:
G_0 and $G_1$are two isomorphic graphs.
$G_0 =< N_0, E_0 >$, $G_1 =< N_1, E_1 >$,
with N being the number of vertices and E the number of edges,
with $\mid N_0 \mid = \mid N_1 \mid$ and $\mid E_0 \mid = \mid E_1 \mid$
$G_0 \approx G_1$ (G_0 and G_1 are isomorphic) $\Leftrightarrow \exists \pi$ (mapping/permutation)
$\forall u, v$ such that $[u, v] \in E_0$
$[\pi(u), \pi(v)] \in E_1$
and $\forall u', v'$ such that $[u', v'] \in E_1$
$\left[\pi^{-1}(u'), \pi^{-1}(v')\right] \in E_0$

P wants to prove to V that G_0 and G_0 are isomorphic, equivalently that $x \in L$ (L being the language of pairs (G_0, G_1) of isomorphic graphs).
If $G_0 \approx G_1$, V will accept
If $G_0 \not\approx G_1$, V will reject.

We define
$\forall x \in L$, $\Pr[V$ accepts x after interacting with $P\ ] \geq \frac{2}{3}$
$\forall x \notin L$, $\forall P'$, $\Pr[V$ accepts x after interacting with P' $] \leq \frac{1}{3}$, P' being any (dishonest) prover.

At the outset of the protocol, P finds a permutation π such that $G_1 = \pi(G_0)$.
P sends π to V. V checks that $G_1 = \pi(G_0)$ and accepts if $G_1 = \pi(G_0)$, rejects otherwise.
This is a problem in **NP**.

Now, consider the *GRAPH-NONISOMORPHISM* problem:
P wants to prove to V that $(G_0, G_1) \in$ *NON-ISO*, equivalently that $x \in L$ (L being the language of pairs (G_0, G_1) of non-isomorphic graphs).

Note:

If V was deterministic, we would have the following:

$\forall x \in L$, $\Pr[V$ accepts x after interacting with $P\ ] = 1$

$\forall x \notin L$, $\forall P'$, $\Pr[V$ accepts x after interacting with $P'\ ] = 0$ (V will always reject since no proof exists!)

This defines **NP** .

At the outset of the protocol, V picks a random bit b and a random permutation π. V sends $G = \pi(G_b)$ to P. P finds b' such that $G \approx G_{b'}$ and sends b' to V. If $b = b'$, V accepts. Otherwise V rejects.

So we have:

if $G_0 \not\approx G_1 \Rightarrow \Pr[V \text{ accepts}] = 1$

if $G_0 \approx G_1 \Rightarrow \Pr[V \text{ accepts}] = \frac{1}{2}$

By running it one more time, we get:

$G_0 \not\approx G_1 \Rightarrow \Pr[V \text{ accepts both times}] = 1$

$G_0 \approx G_1 \Rightarrow \Pr[V \text{ accepts both times}] = \frac{1}{4} \leq \frac{1}{3}$

So now we do satisfy the definition we started with.

In general:

$$\left.\begin{array}{c} \geq \frac{2}{3} \\ \leq \frac{1}{3} \end{array}\right\} \Rightarrow \left.\begin{array}{c} \geq 1-\epsilon \\ \leq \epsilon \end{array}\right\} \forall \epsilon > 0 \ (\ \epsilon = \frac{1}{poly(|x|)}.\ \text{Actually } \epsilon = O(\frac{1}{exp(|x|)})\ )$$

$$\Rightarrow \left.\begin{array}{c} \geq 1 - O(exp(-n)) \\ \leq O(exp(-n)) \end{array}\right\}$$

If you run this n times ($n \leq poly(|\ x\ |)$) :

$$\left.\begin{array}{c} \geq \frac{2}{3}, \leq \frac{1}{3} \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ \geq \frac{2}{3}, \leq \frac{1}{3} \end{array}\right\} n$$

V accepts if at least $\frac{n}{2}$ runs are accepted, rejects otherwise.

Let us rewrite our notation of probabilities in a more compact way:

$\forall x \in L$, $\Pr[(P,V)(x)$ accepts $] \geq \frac{2}{3}$

$\forall x \notin L$, $\forall P'$, $\Pr[(P',V)(x)$ accepts $] \leq \frac{1}{3}$

In reality, P doesn't have infinite power.

It has been proven by Shamir that **IP** $=$ **PSPACE**

1.1.2 Interactive Proofs (IP) and Zero-Knowledge (Z-K)

In IP, we are interested in the bahaviour of P (V is fixed). In Z-K, we are interested in the bahaviour of V (we assume P is honest).

In Z-K, when running the protocol (after V is convinced), we don't want the verifier to have anything to prove to a third party that the conversation between him and the prover took place.
More precisely, a protocol is Z-K if V does not gain *anything* from the process of interaction with P and that V obtains *only* the conviction that P knows the secret (i.e. the proof).
So an interactive proof $[P,\ V]$ is *Zero-Knowledge* if *V'* is able to simulate his entire conversation (defined as $VIEW[P,\ V'](x)$, i.e. the transcript of the protocol) with P , without in reality ever talking to P.
More precisely, $\forall V', \exists S_{v'}$ (S is a simulator, probabilistic poly-time algorithm) $\forall x \in L$, such that $VIEW[P, V'](x) = S_{v'}(x)$ (this is equivalent to saying that P doesn't reveal anything to V'). $VIEW[P, V'](x)$ and $S_{v'}(x)$ are random variables.

2 Lecture 2

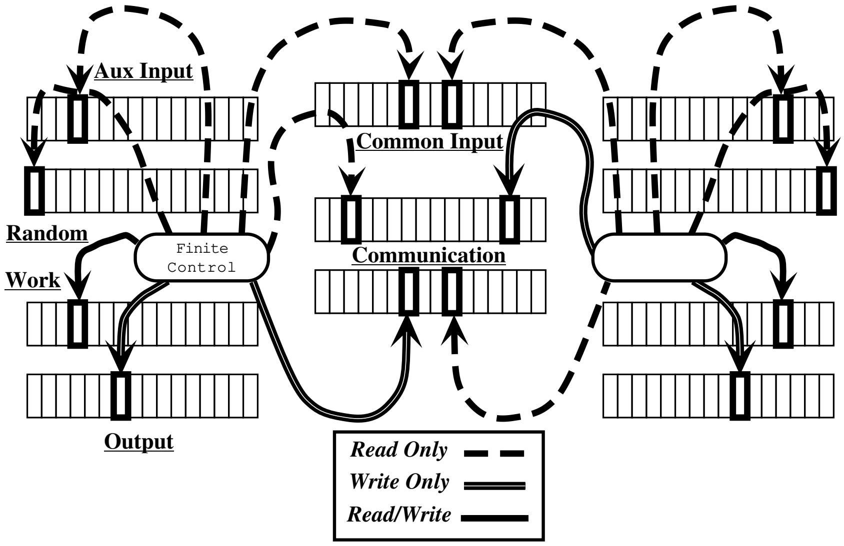
Consider the language L of all numbers which are the product of two primes. P wants to prove to V that $n = p * q$, (p, q: primes).

At the outset of the protocol, P sends p and q to V. V checks that p is prime, q is prime and that $n = p * q$. V accepts if all three checks succeed, rejects otherwise.
This protocol is NOT *Zero-Knowledge* because *FACTORING* is hard.

Now, consider the same language L of numbers $n = p * q$ (p, q: primes). The following protocol achieves the same thing as the previous one except that the factorization is not in the conversation.

At the outset of the protocol, V (secretly) picks $r \in Z_n^*$ and sends x to P such that $x = r^2 mod\,(n)$. P computes $\sqrt{x}\, mod\,(n) = w$ and sends back w to V. V checks if $w \neq \pm r$ and if yes, factors n: $n = p * q$. If p is prime, q is prime, $n = p * q$, then V accepts, else V rejects.
This protocol is also NOT *Zero-Knowledge*.
One can think that a simulator can do this by setting $x = y^2 mod\,(n)$ (second step) and $w = random\, y$ (first step). But this cannot work because we cannot know r; "we have to keep track of V's mind".

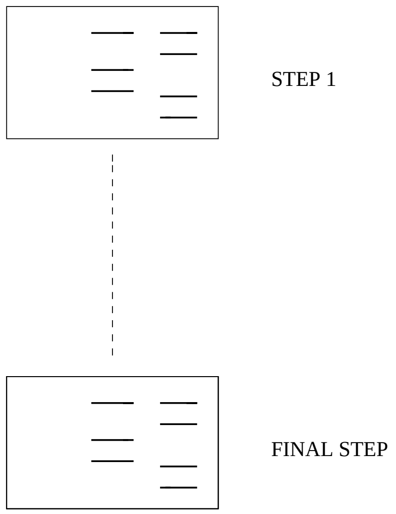
To be more precise about the *Zero-Knowledge* notion (so that no protocol can verify the *Z-K* conditions without truly being *Z-K*), let us consider P and V to be Turing machines with the following tapes and communication scheme:



where the dashed arrows are "read-only", the white arrows "write-only" and the black arrows "read and/or write".

The *VIEW* of a verifier V is the sequence of states and tape contents (Random tape, Work tape, Common Input tape, Communication tape, Output tape, Input tape) from the beginning to the end of a computation.

$\forall V'$(probabilistic polynomial), $\exists S_{v'}$(probabilistic polynomial), $\forall x \in L$,
$VIEW_{v'}[P,\ V'](x) = S_{v'}\,(x)$



About the Auxiliary Input (AI) tape:
"if the verifier knows nothing about proving $x \in L$, then he learns nothing more about it".
So if he increases his knowledge (AI), it should still be *Z-K*.
"if the verifier knows something about proving $x \in L$, then he learns nothing more about it".
So $\forall V'$ (prob. poly) $\exists S_{v'}$ (prob. poly) $\forall x \in L, \forall AI$
$VIEW[P,\ V'(AI)](x) = S_{v'}\,(AI)\,(x)$.
(So regardless of the prior knowledge of the verifier, he doesn't acquire more knowledge for proving membership by talking to the prover).

Let's look at an example:
Consider the *GRAPH-ISOMORPHISM* problem. The prover P wants to prove that the graphs G_0 and G_1 are isomorphic.

At the outset of the protocol, P randomly chooses a bit $b \in_R \{0,1\}$ ($\in_R$ = random with uniform distribution) and a permutation $\pi \in_R S_n$ (S_n is the set of permutations of {1,2,...,n}).
P computes $G = \pi\,(G_b)$ and sends it to V. V randomly picks $b' \in_R \{0,1\}$ and sends it to P (V wants to see $G \approx G_{b'}$). P checks if $b = b'$. If they are equal, he sends $\sigma = \pi$ to V. Otherwise, he sends $\sigma = \pi \circ \pi_{b'b}$ (where $\pi_{b'b}$ maps $G_{b'}$ to

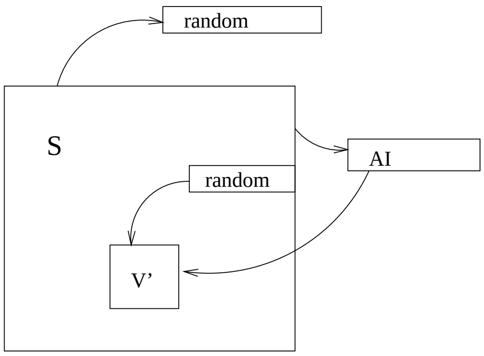
G_b) to V. V checks if $G = \sigma G_{b'}$. If they are equal, V accepts, otherwise V rejects.
So we have:

if $G_0 \approx G_1 \Rightarrow$ Pr($[P, V]$ (G_0, G_1) = accept)=1
if $G_0 \not\approx G_1 \Rightarrow$ Pr($[P\text{'}, V]$ (G_0, G_1) = accept)$\leq \frac{1}{2}$ (by doing this twice, we get $\frac{1}{4}$ and therefore, we do obey the IP properties).

So those 2 properties tell us that this is an *Interactive Proof System.*
Moreover, it is a *Zero-Knowledge Proof System.*

This protocol is *Z-K*:
$\forall V', \exists S_{v'}\ \forall x, \forall AI\ \ VIEW[(P,\ V\text{'}(AI))(x)] = S_{v'}\,(AI)\,(x)$

Program for $S_{v'}$:



- Copy fresh random bits and AI to *V'* random tape and AI tape
- Pick a random bit c
- Pick $\rho \in_R S_n$ (a random permutation)
- $G' = \rho(G_c)$
- Allow *V'* to run on input (G_0, G_1) until *V'* answers after seeing *G'*.
- *V'* replies *b'*.
- if $b' = c$ (prob=$\frac{1}{2}$), then proceed as P and send ρ to *V'* and complete simulation.
- if $b' \neq c$ (prob=$\frac{1}{2}$), then start again.

(What happens here is that upon failure, you start again ignoring the failures).

3 Lecture 3

Comments on Simulator for ISO:

- Whenever we have polynomial time V, we have polynomial time S;
- S can run V', but S doesn't know what V' does;
- G is uniformly distributed among the isomorphic copies of G_0;
- The distribution of the VIEWs of ISO is exactly the same as the distribution of simulator for ISO;
- Running time of S is EXPECTED polynomial.

Example 3.1 *IP for Graph Non-Isomorphism:*

$$
\begin{array}{lll}
 & G_0 \not\approx G_1 & \\
\underline{P} & (on\ n\ nodes) & \underline{V} \\
 & & b \in_R \{0,1\}, \\
 & & \Pi \in_R S_n, \\
 & & G = \Pi(G_b) \\
 & \xleftarrow{G} & \\
b' = (G \approx G_0\ ?\ 0 : 1) & & \\
 & \xrightarrow{b'} & \\
 & & \textbf{if } (b = b') \\
 & & \textbf{then } \textit{accept} \\
 & & \textbf{else } \textit{reject}.
\end{array}
$$

This proof is NOT ZK.

Example 3.2 *ZK proof for Graph Non-Isomorphism:*

$$
\begin{array}{lll}
 & G_0 \not\approx G_1 & \\
\underline{P} & (on\ n\ nodes) & \underline{V'} \\
 & & b \in_R \{0,1\}, \\
 & & \Pi_0, \Pi_1 \in_R S_n, \\
 & & G'_0 = \Pi_0(G_b), G'_1 = \Pi_1(G_{\bar{b}}) \\
 & \xleftarrow{G'_0, G'_1} & \\
loop: & & \\
 & & b_i \in_R \{0,1\}, \\
 & & \Pi_0^i, \Pi_1^i \in_R S_n, \\
 & & G_0^i = \Pi_0^i(G_{b_i}), G_1^i = \Pi_1^i(G_{\bar{b_i}})
\end{array}
$$

$$\xleftarrow{G_0^i, G_1^i}$$

$c_i \in_R \{0, 1\}$

$$\xrightarrow{c_i}$$

$$V' \ sends \ back$$
$$(G_0, G_1) \approx (G_0^i, G_1^i) \ if \ c_i = 0$$
$$(G'_0, G'_1) \approx (G_0^i, G_1^i) \ if \ c_i = 1$$
$$\longleftarrow$$

$b' = (G'_b \approx G_0 \ ? \ 0 : 1)$

$$\xrightarrow{b'}$$

if $(b = b')$
then *accept*
else *reject*.

<u>Simulator</u> for V:

On input (G_0, G_1),

1. Run V until it sends (G'_0, G'_1);
2. Run V until it sends (G_0^i, G_1^i);
3. Send $c_i = 0$ to V and get $(G_0, G_1) \approx (G_0^i, G_1^i)$;
4. Restart and run V until it sends (G'_0, G'_1);
5. Run V until it sends (G_0^i, G_1^i);
6. Send $c_i = 1$ to V and get $(G_0, G_1) \approx (G'_0, G'_1)$;
7. Compute $(G_0, G_1) \approx (G'_0, G'_1)$;
8. Do more runs of V by picking at random (1-3) or (4-6);
9. Finally answer b' to V and leave V finish its calculation.

4 Lecture 4

4.1 Simulator for Graph Non-Isomorphism

On input $G_0, G_1, (AI, R), K$,

- * let V' run until it sends $G_0^{'}, G_1^{'}$,
- For $i = 1$ to k do
  - let V' run until it sends $G_0^{'}, G_1^{'}$,
  - set $C_i = 0$ and send to V',
  - "try" to get $(G_0, G_1) \cong (G_0^i, G_1^i)$,
  - reset V' as in *,
  - set $C_i = 1$ and send to V',
  - "try" to get $(G_0^{'}, G_1^{'}) \cong (G_0^i, G_1^i)$,
  - if both succeed then compute $(G_0, G_1) \cong (G_0^{'}, G_1^{'})$,
  - flip a random coin C and continue conversation with $C_i = C$(but just stop if $C_i = C$ leads to failure) and koutput the partial view.

  where k is a part of the input, a function of number of vertices. With probability $\geq 1 - 2^{-k}$, simulator has computed $(G_0, G_1) \cong (G_0^{'}, G_1^{'})$
- ** send to V' the bit b' such that $G_0 \cong G_{b'}^{'}$,
- let V' finish its calculation and output the view.

Claim 4.1 *This simulator produces valid views except with probability* $2^{-(k+1)}$. *The simulator runs in worst case probalistic polynomial time.*

4.2 Indistinguishability

We give some definitions that will be useful from here on.

For the three following definitions, let X_n and Y_n be families of distributions.

Definition 4.2 (Absolute Indistinguishability) X_n *is absolutely indistinguishable from* Y_n *if*

$$\forall_f \ \forall_n \ Pr(f(X_n) = 1) = Pr(f(Y_n) = 1)$$

Definition 4.3 (Statistical Indistinguishability) X_n *is statistically indistinguishable from* Y_n *if*

$$\forall_f \ \forall_{P:polynomial} \ \forall_{n \geq n_0} \ |Pr(f(X_n) = 1) - Pr(f(Y_n) = 1)| \leq 1/P(n)$$

Definition 4.4 (Computational Indistinguishability) *(or Polynomial-Statistical Indistinguishability)*
X_n is computationally indistinguishable from Y_n if

$$\forall_{A\text{:}fast\ algo.}\ \forall_{P\text{:}polynomial}\ \forall_{n\geq n_0}\ |Pr(A(X_n)=1)-Pr(A(Y_n)=1)|\leq 1/P(n)$$

If in general, we set $k = |G_0 G_1|$, some measure of the size of these graphes. We can set $k = n(\#of vertices)$. Thus the running time is $poly(n)$, and the error probability $\leq 2^{-(n+1)} \leq \frac{1}{poly(n)}, \forall poly, \forall n$large enough .

We can improve above simulator from statistical Zero-Knowledge to Perfect Zero-Knowledge simply be calculating the isomorphism by ourselves at step ** if fail finding $G_0 \cong G_{b'}^{'}$ in expected time, and return the right answer. The time to calculate the isomorphism is exponential, but the probability to really calculate is $\leq \frac{1}{2^{n+1}}$. Therefore, the expected running time is still in $O(1)$.

If we have a perfect Zero-knowledge proof for some problem, then we will always have a statistical Zero-knowledge proof for its complement.

5 Lecture 5

5.1 Bit Commitment

Property

- **binging property** there is a unique value b to which Alice is commited, i.e, Alice cannot change her mind in the middle.
- **concealing property** Bob is unable to tell the value of the b until the unveil phrase.

Example 5.1 *Computational conceiling and perfectly binding:*
The assumption is the polynomial V cannot distinguish isomorphic copies of G_0 and G_1 from each other.
Let $G_0 \not\cong G_1$

$$\begin{array}{rcl}
Alice & & Bob\ (may\ work\ offline) \\
\textbf{Commit}\ to\ b: & & \\
\Pi \leftarrow_R S_n, & & \\
G \leftarrow \Pi(G_b). & & \\
& \xrightarrow{G} & \\
\textbf{Unveiling}: & & \\
& \xrightarrow{b,\Pi} &
\end{array}$$

Example 5.2 *Computational Zero-Knowledge Proof for 3-COL problem*
We will use 2-bit to commit the color, black$=00$, white$=10$ and red$=01$.

Zero-Knowledge Proof $G \in 3 - COL$

- *let G be the graph and let C be a $3 - CORLORING$ of G,*
- *let B, R, W be the colors of the $3 - COLORING$,*
- *P picks at random a permutation $\{B, R, W\}$ and creats a $3-COLORING$ identical to C with the permuted colors,*
- *P commits tp the colors by a bit commitment scheme (Verifier cannot ktell what the colors are),*
- *V selects at random one edge of the graph G (2 neighboring regions),*
- *P unveils the colors corresponding to the chosed edge,*
- *V accepts if the two color are different.*

First, we want to prove this is Interactive Proof for $3-COL$:
If $G \in 3-COL$, $Pr[(P,V)(G)accepts] = 1$,
If $G \notin 3-COL$, there is at least one edge wrong, $\forall P'$,

$$Pr[(P',V)(G)accepts] \leq 1 - \frac{1}{E}$$

where E is the number of edges in G, and $E = O(n^2)$.

If we repeat k times and accept iff V accepts k times, the error probability should be $\leq (1-1/E)^k$.

If we set $E = \theta(n^2)$, and $k = \theta(n^2)$, then

$$Pr[error] \leq (1-1/E)^k \approx (1-\frac{1}{n^2})^{n^2} \approx 1/e$$

If we set $k = \theta(n^3)$, we can have the error probability decreases exponentially fast.

Next, we want to prove the Zero-Knowledge:
Simulator on input G,

- *pick an edge at random,*
- *picks at random two colors from B, R, W,*
- *colors that edge accordingly,*
- *colors everyhting else of the 3rd color,*
- *commits to this "coloring" and asks V' to choose an edge,*
- *if edges are the same, continue, else repeat*

The expected running time is $O(n^5)$.

Example 5.3 *Quadratic Non-Residues:*
$y \in QNR_n$
$\nexists x$ *such that* $x^2 \equiv y(\ mod\ n)$

$\underline{Prover}$	n	$\underline{Verifier}$
Commit $to\ b$:		
$pick\ random\ r \in Z_n^*$,		
	$\xrightarrow{Z=r^2y^b mod\ n}$	
Unveiling :		
	$\xrightarrow{b,\ r}$	
		$verify\ Z = r^2y^b mod\ n$

Example 5.4 *Bit Commitment from One-Way Permutations:*
$RSA_n(x) = x^e \; mod \; n,$
$where \; n = p * q, \; \phi(n) = (p-1)(q-1), \; gcd(e, \phi(n)) = 1.$

$$\begin{array}{rcl}
\underline{Prover} & n, e & \underline{Verifier} \\
\textbf{Commit } to\ x : & & \\
 & \xrightarrow{Z = x^e \; mod \; n} & \\
\textbf{Unveiling} : & & \\
 & \xrightarrow{x} & \\
 & & verify\ Z = x^e \; mod \; n \\
We\ commit\ to\ b\ rather\ than\ x : & & \\
pick\ random\ x\ such\ that\ LSB(x) = b, & & \\
 & \xrightarrow{Z = x^e \; mod \; n} & \\
\textbf{Unveiling} : & & \\
 & \xrightarrow{x,\, b} & \\
 & & verify\ Z = x^e \; mod \; n
\end{array}$$

The assumption is that it is difficult to determine LSB(x) given $x^e \; mod \; n$. This assumption is true. $\Longleftrightarrow$ RSA is difficult to invert.

6 Lecture 6

6.1 Bit Commitments using Hard Core Predicates

What is a "hard core predicate" ?

Let X be an input string and let w be a binary bit.
Define a function , f , such that $f(X)=w$. We say w is a hard core predicate if no polynomial time algorithm can predict it's value with probability $> \frac{1}{2} + \frac{1}{poly(|X|)}$.

6.1.1 Examples of bit commitments using hard core predicates:

6.1.2 1) Using the RSA cryptosystem :

The following is a bit commitment protocol between a prover P and a verifier V over a bit b.

Public: n, $a \in Z_n^*$
Secret: $n = p * q$ where p and q are prime

P		V
pick $m \in_R Z_n^*$ such that the (least significant bit of m)$= b$ i.e. $lsb(m)=b$	$c = m^a \ mod \ n$ $\longrightarrow$	
Unveil	m , b $\longrightarrow$	check: $lsb(m)=b$ $c = m^a \ mod \ n$

Notes:

Choosing m such that $m \in Z_n^*$ will make the RSA function a one-way permutation over the elements of Z_n^* .
$lsb(m)$ is considered unpredictable given $m^a \ mod \ n$, a and n i.e. no polynomial

time algorithm can guess the predicate with
probability $> \frac{1}{2} + \frac{1}{poly(|m|)}$.

Predicting $lsb(m) \Leftrightarrow$ Computing m (known to be intractable given
. $m^a \ mod \ n$, a and n)

6.1.3 2) Using Rabin's cryptosystem :

The following is a bit commitment protocol between a prover P and a verifier V over bit b.

Public: n
Secret: $n = p * q$ where p and q are primes such that $p \equiv q \equiv 3 \ mod \ 4$

P		V
pick $m \in QR_n$ such that $lsb(m)=b$ and $r^2 \equiv m \ mod \ n$	$c = m^2 \ mod \ n$ $\longrightarrow$	
Unveil	m , b , r $\longrightarrow$	check: $lsb(m)=b$ $c = m^2 \ mod \ n$ $m \in QR_n$ $r^2 \equiv m \ mod \ n$

Notes:

Choosing m such that $m \in QR_n$ will make Rabin's function a one-way permutation over the quadratic residues of Z_n.

lsb(m) is considered unpredictable given m^2 mod n and n i.e. no polynomial time algorithm can guess the predicate with probability $> \frac{1}{2} + \frac{1}{poly(/m/)}$.

Predicting *lsb(m)* $\Leftrightarrow$ Computing m (known to be intractable given
. m^2 mod n and n)

6.1.4 3)Using the cryptosystem based on the Discrete Logarithm Problem:

The following is a bit commitment protocol between a prover P and a verifier V over a bit b.

Definition :

$$half(x) = \begin{cases} 0, x \leq \frac{p-1}{2} \\ 1, x > \frac{p-1}{2} \end{cases}$$

where p is prime

Public: p which is prime, g which is a primitive element of $\in Z_p^*$

P		V
pick $x \in_R Z_p^*$ such that *half(x)=b*	$\xrightarrow{c = g^x \ mod \ p}$	
Unveil .	$\xrightarrow{x\ ,\ b}$	check: *half(x)=b*

. $c = g^x \ mod \ p$

Notes:

Choosing x such that $x \in Z_p^*$ will make the function (which raises g to power x) a one-way permutation over the elements of Z_p^* .
half(x) is considered unpredictable given $g^x \ mod \ p$, g and p i.e. no polynomial time algorithm can guess the predicate with
probability $> \frac{1}{2} + \frac{1}{poly(|x|)}$.

Predicting *half(x)* $\Leftrightarrow$ Computing x (known to be intractable given
. $g^x \ mod \ p$, g and p)

- **The previous examples lead us to an important result...**

Let π (m) be a one-way permutation.

Consider the following:

π' *(m,r)* = [π *(m)*, r]

where m is a given input and r is a random input such that $\|r| = |m|$.

Note that π' (m , r) is also a one-way permutation .

Theorem by Goldreich and Levin :

The predicate , (m $\bigodot$ r) , defined as:

(m $\bigodot$ r) = ($\sum m_i r_i$) mod 2

is a *hard core predicate* of π' (m , r) .

Bit Commitment of b :

P		V
pick $r \in_R \{$ input set $\}$ such that *(m* $\bigodot$ *r)= b*	c=π' *(m,r)* = [π *(m), r*] $\longrightarrow$	
Unveil	m , b , r $\longrightarrow$	check: *(m* $\bigodot$ *r)= b* c=π' *(m,r)*

Notes:

Predicting (m $\bigodot$ r) $\Leftrightarrow$ Computing π^{-1}*(m)* (known to be intractable given π' (m , r) for random r since π is a one-way prem.

References

[BB88] Pierre Beauchemin and Gilles Brassard. A generalization of hellman's extension of shannon's approach to cryptography. In Carl Pomerance, editor, *Proc. CRYPTO 87*, pages 461–461. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[BS93] E. Biham and A. Shamir. *A Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[dB88] Bert den Boer. Cryptanalysis of F.E.A.L. In Christoph G. Günther, editor, *Advances in Cryptology—EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 293–299. Springer-Verlag, 25–27 May 1988.

[Hel80] M. E. Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26:401–406, 1980.

[Mat94] M. Matsui. Linear cryptanalysis method for DES cipher. *Lecture Notes in Computer Science*, 765:386–397, 1994.

[Mur90] S. Murphy. The cryptoanalysis of FEAL-4 with 20 chosen plaintexts. *Journal of Cryptology*, 2:145–154, 1990.

[MVV97] A. J. (Alfred J.) Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography.* The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[NBS93] Data encryption standard (des). Technical Report FIPS PUB 46-2, National Bureau of Standards, 1993.

[NIS80] Des modes of operation. Technical Report FIPS PUB 81, National Institute of Standards and Technology, december 1980.

[NIS99] http://www.nist.gov/aes/, 1999.

[Sha48] C. E. Shannon. A mathematical theory of communication. volume 27, pages 379–423, 623–656, 1948.

[Sta99] William Stallings. *Cryptography and network security: principles and practice.* Prentice-Hall, Inc., Upper Saddle River, NJ 07458, USA, second edition, 1999.

[Sti95] Douglas R. Stinson. *Cryptography Theory and Practice.* CRC Press, Boca Raton, 1995.

[VK84] V. L. Voydock and S. T. Kent. Security mechanisms in a transport layer protocol. *Computer Networks*, 8:433–450, 1984.