CSL 759 – Cryptography and Network Security Jan 15, 2014

Lecture 5

Instructor: Shweta Agrawal *Scribe: Karan Goel*

1 Computing One-Bit

Lemma 1 *Given $g^x \bmod p \in \mathbb{Z}_p^\star$, we can compute the LSB of x efficiently.*

Proposition 1 *Consider a quadratic equation,*

$$x^2 \equiv a \bmod p$$

in the group $\mathbb{Z}_p^\star$.

Claim: This equation has exactly 0 or 2 roots, and it cannot have 1 root. We prove this as follows.

Proof. For this equation to have a single root is impossible as $x \neq -x \bmod p$, and also

$$x^2 \equiv y^2 \bmod p$$
$$(x+y)(x-y) \equiv 0$$
$$x = \pm y$$
$$\because \mathbb{Z}_p \text{ is in integral domain}$$

Thus, the equation cannot have more than two roots as well. ■

Definition 1 (Quadratic Residues) *$a \in \mathbb{Z}_p^\star$ is called a quadratic residue if*

$$x^2 \equiv a \bmod p$$

has exactly two solutions.

We can use the Quadratic Residue (QR) to determine the $LSB(x)$. However, it can get hard to determine QRs in a composite order field.

Lemma 2 *Suppose g is a generator of $\mathbb{Z}_p^\star$, $a \in \mathbb{Z}_p^\star$ and $a = g^z$, then*

$$a \text{ is a } QR \Leftrightarrow z \text{ is even} \Leftrightarrow a^{\frac{p-1}{2}} \equiv 1 \bmod p$$

Proof.

1.

$$\text{If } z = 2w,$$
$$\implies a = g^z = (g^w)^2 \text{ is a QR.}$$

Conversely,

$$\text{if } a = (g^w)^2,$$
$$\text{then } z \equiv 2w \bmod (p-1),$$
$$\text{and since } (p-1) \text{ and } 2w \text{ are even}$$
$$\implies 2w \bmod (p-1) \text{ is even}$$
$$\implies z \text{ is even}$$

2.

$$\text{If } z = 2w,$$
$$\implies a^{\frac{p-1}{2}} = g^{\frac{z(p-1)}{2}} = g^{w(p-1)} \equiv 1 \bmod p \text{ (By Fermat's Little Theorem)}$$

Conversely,

$$\text{if } g \text{ is a generator and } g^{\frac{z(p-1)}{2}} \equiv 1$$
$$\implies z \cdot \frac{p-1}{2} \equiv 0 \bmod (p-1)$$
$$\implies z \cdot \frac{p-1}{2} = w(p-1), \text{ , for some } w$$
$$\implies z = 2w$$
$$\therefore z \text{ is even}$$

■

Corollary: Exactly half of the elements in $\mathbb{Z}_p^\star$ are QR.

Definition 2 (Legendre Symbol) *Suppose p is prime and $a \in \mathbb{Z}_p^\star$,*

$$(\frac{a}{p}) \equiv a^{\frac{p-1}{2}} \bmod p,$$
$$(\frac{a}{p})^2 \equiv 1 \bmod p, (\frac{a}{p}) \in \{+1, -1\},$$

where $(\frac{a}{p})$ is called the Legendre symbol of a, and a is a QR if and only if its Legendre symbol is $+1$.

Note that the Legendre symbol can be computed efficiently. Thus, we can find the LSB easily (z will become even/odd), depending on the Legendre symbol.

There might be a possibility that a part of x is discovered which if it is being used as a secret key, can be dangerous.

A **trapdoor permutation** can be inverted using some secret information.
If we have a group $\mathbb{Z}_n^\star$, where $n = pq$ (p and q are arbitrary large primes), pick $e \in \mathbb{Z}_{\phi(n)}^\star$.
$\exists$ some d such that $e.d = 1 \bmod \phi(n)$ ($\because$ inverses in a group).

The RSA Encryption Scheme

Let the Public Key be (n, e) and the Secret Key be d. Let there be a OWF, $f(x)$ such that

$$f(x) = x^e \bmod n$$

Using the SK d, one can quickly recover x, as shown below

$$(x^e)^d = x^{ed \bmod \phi(n)} = x^1 = x$$

1. Security is only proven for a uniformly random distribution of x (in which case $f(x)$ is hard to invert). However, messages are not guaranteed to be uniformly random. Therefore, RSA is secure only if messages are uniformly random.

2. The RSA scheme does not guarantee that every bit of x is kept secret. It is possible that a bit of x may be discovered, which can potentially leak some partial information.

If factoring is easy, then RSA is easy, and the encryption scheme will no longer be secure. However, the converse is not true and RSA is not as hard as factoring.

Chinese Remainder Theorem

Theorem 3 (Chinese Remainder Theorem) *Let m_1, m_2,..., m_k be pairwise relatively prime and let $m = \Pi m_i$. Then $\forall a_1 \in \mathbb{Z}_{m_1}$, $\forall a_2 \in \mathbb{Z}_{m_2}$, . . ., $\forall a_k \in \mathbb{Z}_{m_k}$, $\exists$ a unique $a \in \mathbb{Z}_m$ such that*

$$a \equiv a_i \bmod m_i, \forall i \in 1, .., k$$

Proof. The proof is left as an exercise for the reader.

■

Definition 3 (Jacobi's Symbol) *If $n = pq$, where p, q are 2 primes, then we define the Jacobi's symbol of $a \in \mathbb{Z}_n{}^*$, where $a = (a_1, a_2)$, to be*

$$\left(\frac{a}{n}\right) = \left(\frac{a_1}{p}\right)\left(\frac{a_2}{q}\right)$$

where the RHS contains Legendre symbols (prime order components) that are multiplied together. There exists a PPT algorithm to compute $\left(\frac{a}{n}\right)$*, without factoring* n *into its primes* p *and* q*.*

Lemma 4 *If* a *is a QR, and* $\left(\frac{a}{n}\right)$ *is the Jacobi's symbol of* a*, then* $\left(\frac{a}{n}\right) = +1$*. The converse is not necessarily true.*

Given an element, determining the QR over composite order groups is a hard problem.

CSL 759 – Cryptography and Network Security Jan 17,2014

Lecture 6: Quadratic Residues and Hardcore Bits

Instructor: Shweta Agrawal *Scribe: Dhiraj Madan*

1 QR Assumption and One wayness of SQUARE

Definition 1 ***Jacobi Symbol****:If $a \in \mathbb{Z}_n^*$,where n=pq,p and q are primes, and $a \equiv a_1 \ mod \ p$, $a \equiv a_2 \ mod \ q$ then Jacobi symbol of a, $\left(\frac{a}{n}\right) = \left(\frac{a_1}{p}\right)\left(\frac{a_2}{q}\right)$ where $\left(\frac{a_1}{p}\right)$ and $\left(\frac{a_2}{q}\right)$ are legendre symbols of a_1 and a_2 with respect to p and q respectively.*

Recall that a is a quadratic residue over $\mathbb{Z}_p^*$ (where p is a prime)iff Legendre Symbol=+1. We also claim without proof that it is easy to compute square roots of a in $\mathbb{Z}_p^*$.

Lemma 1 *If a is a QR in $\mathbb{Z}_n^*$,(where n=pq,p and q are primes)then $\left(\frac{a}{n}\right)$=+1.*

Proof. If $a \equiv x^2 \ mod \ n \ for \ some \ x \in \mathbb{Z}_n^*$
$\therefore a \equiv x^2 \ mod \ pq$
$\therefore pq|a - x^2$
$\therefore p|a - x^2 \ and \ q|a - x^2$
$\therefore a \equiv x^2 \ mod \ p \ and \ a \equiv x^2 \ mod \ q$
$\therefore \left(\frac{a}{p}\right) = +1 \ and \ \left(\frac{a}{q}\right) = +1$
$\therefore \left(\frac{a}{n}\right) = +1$

■

However the converse of the above lemma is not true.
QR Assumption: For randomly chosen n=pq,random a$\in \ I_n = \left\{b : \left(\frac{b}{n}\right) = +1\right\}$,
for any PPT A,
$\Pr\{A(a) \rightarrow \alpha \ \wedge \ \alpha = QR(a)\} \leq \frac{1}{2} + negl(n)$.
In words,the above means that there is no algorithm to check whether a random number in $\mathbb{Z}_n^*$ is a quadratic residue,with a probability significantly better than that of flipping a coin.
Exercise: Given p and q and n=pq,how to compute quadratic residue?
(**Hint:**Use Chinese Remainder Theorem).
From the above exercise it is clear that if factoring is easy than inverting SQR is also easy.Now we claim that the converse is also true.

Theorem 2 *If SQR is easy to invert then factoring is easy.*

Proof. Assume SQR is easy to invert with a non negligible probability ϵ(no. of bits). We now factor n with probability ϵ(no. of bits)/2.
Algorithm:

1. Choose a random x $\in \mathbb{Z}_n^*$.
2. Choose a=x^2mod n.
3. Let b=SQ^{-1}(a).
   If b=$\pm$x,fail and exit.
4. If b$\neq \pm x$,
   Output gcd(x+b,n) as a factor and n/gcd(x+b,n) as the other.

Analysis: If a$\equiv (a_1, a_2)$mod(p,q), Then by CRT square roots of x^2 are :-
(a_1, a_2),$(-a_1, -a_2)$
$(-a_1, a_2)$,$(a_1, -a_2)$
Now with probability $\epsilon/2$,
we will get the one of the last 2 pairs of factors.
In case b=$(-a_1, a_2)$, x+b=(0,$2a_2$) and hence gcd(n,x+b)=p and n/gcd(n,x+b)=q.
Similarly,in case b=$(a_1, -a_2)$, x+b=($2a_1$,0) and hence gcd(n,x+b)=q and n/gcd(n,x+b)=p.
Thus with probability $\epsilon/2$, we have inverted. If ϵ is non negligible then so is $\epsilon/2$.Hence if finding square roots with respect to $\mathbb{Z}_n^*$ is easy for any composite n, then so is factoring. ■

2 Hardcore Bits

Definition 2 *Hardcore Bits:A function $h : \{0,1\}^* \rightarrow \{0,1\}$ is called a hardcore bit for some function f if:-*

1. *h(x) is easy to compute from x.*
2. *No PPT algorithm can predict h(x) (given f(x)) better than flipping a coin.*

Formally,
$\forall$*PPT A P(A(f(x))*$\rightarrow h(x)|x \leftarrow \{0,1\}^k$*)*$\leq$*1/2+negl(k).*

Note:A function being one way does not imply that all bits are hardcore w.r.t the function.
Consider f , a length preserving OWP,define f'(x,y)=x||f(y)
Clearly no bit in x is a hard core bit for f'
But f' can not be inverted(otherwise f is not one way).

2.1 Constructing hardcore bits for OWF/OWP

There are 2 strategies for constructing examples of hardcore bits:-

1. Choose some concrete example of f say f(x)=g^x mod p.(Where g is a generator)
2. Take an arbitrary OWF,and exhibit general construction of hardcore bits.This is called Goldreich Levin HCB.

Define MSB(x)=

$$f(x) = \begin{cases} 0 & : x < \frac{p-1}{2} \\ 1 & : Otherwise \end{cases}$$

Theorem 3 *MSB(x) is a hardcore bit for g^x mod p.*

Proof. We will show that given an algorithm that always computes HCM=MSB(x),we can construct an algorithm that always inverts g^x mod p.
Define A_{INVERT} as:-
Let x=[x_l,...,x_0] in binary and y=g^x mod p.

1. Extract x_0=LSB(x).
   Set $y \leftarrow \frac{y}{g^{x_0}} = g^{[x_l,\ldots,x_1,0]}$
2. Note that y has 2 square roots :-
   $g^{[x_l,\ldots,x_1]}$ and -$g^{[x_l,\ldots,x_1]}$,which can be computed in polynomial time since p is a prime.
   Observe that $g^{\frac{p-1}{2}}$=-1
   ($\because$ g is a generator
   $\implies$ g^{p-1}=1
   $\implies$ $g^{(p-1)/2}$=$\pm$ 1
   Since g is a generator ord(g)=p-1 and hence $g^{(p-1)/2} \neq +1$
   $\implies$ $g^{(p-1)/2}$=-1).
   $\therefore$ Square roots are $g^{[x_l,\ldots,x_1]}$ and $g^{[x_l,\ldots,x_1]+(p-1)/2}$.
   The square root with MSB=0 is the principal square root.
3. Use A_{MSB} to compute the principal square root and recurse to find entire x bit by bit.
   Thus since by iterating l(no. of bits) times,we can successfully find x,
   Total time needed =l*(time for finding MSB+time for finding square roots),which is clearly a polynomial in l(no. of bits).Thus if $\exists$ a PPT algorithm to compute MSB,we can easily compute discrete logarithm which is hard.
   Thus computing MSB of x given g^xmod p must be hard.
   Thus MSB(x) is a hard core bit for g^xmod p.

3 Summary

1. Checking for quadratic residues and finding square roots is easy with respect to a prime, but is as hard as factoring in $\mathbb{Z}_n^*$ where n is a composite.
2. A function $h : \{0,1\}^* \rightarrow \{0,1\}$ is called a hardcore bit for some function f if:-
   (a) h(x) is easy to compute from x.
   (b) No PPT algorithm can predict h(x) (given f(x)) better than flipping a coin.
3. MSB(x) is a hard core bit for f(x)=g^x mod p.

CSL 759 – Cryptography and Network Security Jan 21, 2014

Lecture 7: Hardcore Bits

Instructor: Shweta Agrawal *Scribe: Ishaan Preet Singh*

1 Recall

Definition 1 *$h : \{0,1\}^* \to \{0,1\}$ is a hardcore bit for a One Way Function f if $h(x)$ is easy to compute given x and $h(x)$ is hard to compute given $f(x)$. Formally, $\forall PPT$ A;*

$$Pr(A : f(x) \leftarrow h(x)) \leq \frac{1}{2} + negligible \tag{1}$$

2 General Hardcore Bit for a OWF

Definition 2 Random Parity : *If $x \in \{0,1\}^k$ and $r \in \{0,1\}^k$ then*

$$h(x,r) = \sum r_i x_i \mod 2 \tag{2}$$

Given a OWF $f : \{0,1\}^k \to \{0,1\}^k$ define a new function $g_f : \{0,1\}^k \to \{0,1\}^k$ such that

$$g_f(x,r) = f(x), r \tag{3}$$

Theorem 1 Goldreich Levin Theorem: *If f is a OWF, the $h(x,r)$ is a hardcore bit for f. Formally, $\forall PPT$ A;*

$$Pr(A : (f(x),r) \leftarrow h(x,r)) \leq \frac{1}{2} + negligible \tag{4}$$

Proof. Assume we have a PPT algorithm that can predict the hardcore bit with high probability, A_{GL} that is,

$$Pr(A_{GL} : (f(x),r) \leftarrow h(x,r)) > \frac{1}{2} + negligible \tag{5}$$

We will now build an algorithm A_{OWF} that will be able to invert f. Initially, we observe that

1. For every r, $< x, r >$ and $< x, r \oplus e_i >$ can together recover x_i, the ith bit of x.
2. We can't test if the algorithm A_{GL} gave us the correct bit but can only give a probability of its correctness. Hence, we can run the algorithm many times and take a majority of the results to get a final answer with a higher probability.

We will not provide a complete proof but we will try and prove the theorem for a subset of our sample set for which we have an inverting function with better chances of success.

Claim 1 *$\exists$ a set* Good $\subseteq \{0,1\}^k$ *with* $|\text{Good}| \geq 2^k \frac{\epsilon}{2}$ *and* $x\ \epsilon$ *where* ϵ *is a non-negligible function and Good if for* x *we have* A_{GL} *such that*

$$Pr(A_{GL} : (f(x), r) \leftarrow h(x, r)) \geq \frac{3}{4} + \frac{\epsilon}{2} \tag{6}$$

Proof. We will now develop a lower bound for the cardinality of Good.
Define $succ(x)$ as $Pr(A_{GL} : (f(x), r) \leftarrow h(x, r))$ For $x\ \epsilon\ Good$, $succ(x) \geq \frac{3}{4} + \frac{\epsilon}{2}$ and let A_{GL} wins mean A_{GL} succeeds in predicting $h(x)$

$$\begin{aligned} Pr(A_{GL}wins) &= Pr(A_{GL}wins|x \in Good)Pr(x \in Good) \\ &\quad + Pr(A_{GL}wins|x \notin Good)Pr(x \notin Good) \\ &\leq Pr(x \in Good) + Pr(A_{GL}wins|x \notin Good) \end{aligned} \tag{7}$$

$$\implies Pr(x \in Good) \geq Pr(A_{GL}wins) - Pr(A_{GL}wins|x \notin Good) \tag{8}$$

$$\implies Pr(x \in Good) \geq \frac{3}{4} + \epsilon - (\frac{3}{4} + \frac{\epsilon}{2}) \tag{9}$$

$$\implies Pr(x \in Good) \geq \frac{\epsilon}{2} \tag{10}$$

$$\implies |Good| \geq \frac{\epsilon}{2} * 2^n \tag{11}$$

We observe that for any i and $x \in Good$ since we know the value $Pr(A_{GL}wins)$ we can calculate that

$$Pr(A_{GL} : (f(x), r) \neq \langle x, r\rangle) \leq \frac{1}{4} - \frac{\epsilon}{2} \tag{12}$$

$$Pr(A_{GL} : (f(x), r) \neq \langle x, r \oplus e_i\rangle) \leq \frac{1}{4} - \frac{\epsilon}{2} \tag{13}$$

Now, to predict the actual ith bit we need to XOR both these bits. The predicted value will be correct either if both are correct or if both are incorrect. Hence the only case in which the predicted bit is incorrect is when only one of them is incorrect.
Hence,

$$Pr(A_{GL}\text{fails on only one of the two}) \leq 2 * Pr(A_{GL}\text{fails on first}) = \frac{1}{2} - \epsilon \tag{14}$$

$$\implies Pr(A_{GL}\text{succeeds to predict the bit}) \geq 1 - (\frac{1}{2} - \epsilon) = \frac{1}{2} + \epsilon \tag{15}$$

■

Now we have A_{GL} that can predict one bit of f with $Pr \geq \frac{1}{2} + non - negligble$ and will construct A'_{OWF} to invert the entire string of bits with a non-negligible probability.

Algorithm: A'_{OWF}

For $i = 1$ to k do

For $j = 1$ to t do

1. Pick r_j at random
2. Run A_{GL} on $f(x, r_j)$ as well as on $f(x, r_j \oplus e_i)$
3. Compute x_{ij} as XOR of the two

Compute $x_i = Majority(x_{ij})$

Claim 2 *If $t = \log(2k)$ then $Pr(x$ computed correctly$) \geq \frac{1}{2}$.*

Proof.

Definition 3 Chernoff Bounds: *If $Z_1, Z_2, \ldots Z_t$ are iid(independent and identically distributed) and $E(Z_i) = \frac{1}{2}, Z = \sum_{i=1}^{t}$ then*

$$Pr(Z < \frac{t}{2}) \leq 2^{-\Omega(t)} \tag{16}$$

Here, Z_i is an indicator variable that x_i is correct.
If $t = \log(2k)$

$$Pr(\text{Majority of bits is incorrect}) \leq \frac{1}{2k} \tag{17}$$

$$\implies Pr(x_i \text{ is wrong}) \leq \frac{1}{2k} \tag{18}$$

$$\implies Pr(\text{Bit string is incorrect}) \leq k * \frac{1}{2k} = \frac{1}{2} \tag{19}$$

$$\implies Pr(\text{Bit string is correct}) \geq \frac{1}{2} \tag{20}$$

■

Now, we've proved that for $x \in Good$, A'_{OWF} inverts f with $Pr \geq \frac{1}{2}$ then

$$Pr(A'_{OWF} \text{ succeeds in inverting} f) \geq Pr(succ | x \in Good) * Pr(x \in Good) \tag{21}$$

$$\implies Pr(A'_{OWF} \text{ succeeds in inverting} f) \geq \frac{1}{2} * \frac{\epsilon}{2} = \frac{\epsilon}{4} \tag{22}$$

$$\tag{23}$$

Hence, we have inverted f with non-negligible probability. ■

CSL 759 – Cryptography and Network Security Jan 21, 2013

Lecture 7: General Hardcore Bit of OWF

Instructor: Shweta Agrawal *Scribe: V.Rajeev*

1 Recap : OWF and hardcore bit

One Way Function: $h : \{0,1\}^k \to 0,1$ is a hardcore bit for a OWF 'f' if
1) h(x) is easy to compute given x
2) h(x) is hard to compute given f(x)
i.e. for any polynomial time algorithm A : $Pr(A(f(x) \to h(x))) \leq \frac{1}{2} + (negligible function)$

Hardcore bit:
given f(x), if MSB(x) can be computed, then entire of $f^{-1}(f(x))$ can be computed.

2 General hardcore bit of a OWF

Definition 1 *For a random parity :if* $x \epsilon 0,1^k and r \epsilon 0,1^k$ *, then* $hx, r = \sum r_i x_i mod 2$

Given OWF $f : \{0,1\}^k \to \{0,1\}^k$,
Define $g_f = 0,1^{2k} \to 0,1^{2k}$ such that: $g_f(x,r) = f(x), r$
(g_f - appendeded function such that inverting $g'_f \equiv inverting' f'$)

3 Goldreich - Levin Theorem

Theorem 1 *If 'f' is a OWF, then h(x,r) is a hardcore bit for* g_f *or more formally for all PPTA,* $P_{x,r}(A(f(x), r) \to h(x, r)) < 1/2 + (negligible function)$

Proof. Contra-positive Method
Assume A_{GL} such that $P_{x,r}(A(f(x), r) \to h(x, r)) \geq 1/2 + (negligible function)$ will build an A_{OWF} that inverts f
Easy Case: Suppose A_{GL} is such that it always computers the hardcore bit. Set r = unit vector & directly recover x each time.
Medium Case: Suppose A_{GL} such that $Pr(A_{GL} succeeds) \geq 3/4 + \varepsilon$, where ε is a non-negligible function
Proof Idea: r needs to be random
(1)Observe that for every$'r', < x, r >$ and $< x, r \oplus e_i >$together recover x_i. Call A_{GL} on $< x, r >$ & $< x, r \oplus e_i >$
Note: Since we can't test when the algo A_{GL} is correct, we run it many times and take majority.
(2) If both answers are same, x_i is obtained.

Proposition 1 ***Claim:*** *there exits a set "GOOD" $\Subset 0,1^k$ such that $|GOOD| \geq 2^n.\varepsilon/2$ and for all $x \epsilon GOOD$: $Pr(A_{GL}wins) \geq 3/4 + (\varepsilon)/2$*
Proof: *Define $succ(x) = Pr(A(f(x), r) =< x, r >)$.*

$$\begin{aligned} &GOOD \text{ is the set of } x \text{ such that: } succ(x) \geq \tfrac{3}{4} + (\varepsilon)/2 \\ Pr_{x,r}(A_{GL}wins) &= Pr(A_{GL}wins|x \in GOOD)xPr(x \in GOOD) + Pr(A_{GL}wins|x \notin \\ &\quad GOOD)xPr(x \notin GOOD) \\ &\leq Pr(x \in GOOD) + Pr(A_{GL}wins|x \notin GOOD) \end{aligned}$$

$$\begin{aligned} Pr(x \epsilon GOOD) &\geq (3/4 + \varepsilon) - (3/4 + (\varepsilon)/2) \\ &= (\varepsilon)/2 \equiv non - negligible \\ &\quad |GOOD| \geq (\varepsilon)/2.x^k \end{aligned}$$

Observe: For any 'i' and x ϵ GOOD

$$\begin{aligned} Pr(A_{GL}(f(x,r)) \neq &< x, r >) \leq 1/4 - (\varepsilon)/2 \\ Pr(A_{GL}(f(x, r \oplus e_i)) \neq &< x, r \oplus e_i >) \leq 1/4 - (\varepsilon)/2 \\ Pr(A_{GL}&\text{fails on atleast on of them}) \leq 1/2 - \varepsilon \\ Pr(A_{GL}&\text{succeeds on both of them}) > 1/2 + \varepsilon \end{aligned}$$

Statement 1: If $x \epsilon GOOD$, and suppose A_{OWF} inverts f with $Pr \geq 1/2$then, objective attained.

$$\begin{aligned} Pr(A_{OWF}&\text{succeeds in inverting f}) \geq Pr(A_{OWF}succeeds|GOOD).Pr(GOOD) \\ &\geq 1/2x(\varepsilon)/2 = (\varepsilon)/4 .....eq^n(2) \end{aligned}$$

Need: A'_{OWF} to invert f' with $Pr \geq 1/2$ for x ϵ GOOD
A'_{OWF}: for i=1 to k do
I) for j=1 to 't'
1) Pick $r_j \leftarrow 0,1^k$
2) Run $A_{GL}(f(x), r_j) as well as A_{GL}(f(x), r_j \oplus e_i)$
3) Compute x_{ij} as XOR of answer.
II) Compute $x_i = majority(x_{ij})$

Proposition 2 ***Claim:*** *if t = log 2k, then Pr(x_i computes correctly) $\geq 1 - 1/2k$*

Lemma 2 *Chernoff: if $z_1, z_2 .... z_t$ are independant & identically distributed and $E(z_i) = 1/2, z = \sum_{i=1-t} z_i$, where z_i is indicator that x_i is correct then, $Pr(z < t/2) \leq 2^t$*

Justifying the Claim: *if t = log2k, Pr(Majority is wrong) $\leq$ 1/2k thus, there exists i, such that x_i computed with A_{GL} is wrong with Pr < k.1/2k So, Pr(A_{GL} is correct for all i) $\geq \frac{1}{2}$*

From Statement 1 and $eq^n 2$, since A'_{OWF} inverts f with $Pr \geq 1/2$ for GOOD x and set GOOD is large enough. ■

CSL 759 – Cryptography and Network Security Jan 22, 2014

Lecture 8: Applications of hardcore bits and Introduction to PRG

Instructor: Shweta Agrawal *Scribe: Ujjwal Kumar Gupta*

1 Applications of Hardcore Bit

1.1 Using hardcore bit for coin tossing on the telephone

How can two parties A and B toss a fair random coin over the phone?

- **Solution-1**
  A tosses the coin and tell the result to B.

  Analysis - If only one of them actually tosses a coin, then person who tosses coin may tell lie.

- **Solution-2**
  Both players toss a coin and they take the XOR as the shared coin

  Analysis -(1) Even if B does not trust A to use a fair coin, he knows that as long as his bit is random, the XOR is also random.
  (2)Whoever reveals his result first has a disadvantage: the other person can adjust his answer to his favor.

- **Solution-3**
  Assume that A and B can not invert OWP then Scheme is as follows:
  1. Alice sends $g_f(x_A,r_A) = f(x_A), r_A$ to Bob.
  2. Bob sends $g_f(x_B,r_B) = f(x_B), r_B$ to Alice.

  3. A sends $x_A \langle x_A,r_A \rangle$
  4. B sends $x_B \langle x_B,r_B \rangle$

  5.A verifies x_B by computing $f(x_B)$ and use $x_B \oplus x_A$ as shared coin.
  6.B verifies x_A by computing $f(x_A)$ and use $x_B \oplus x_A$ as shared coin.

  Analysis -

1. B can verify that x_A is the same as in the first message by applying f_n, therefore A cannot change his result after learning B's result.Similarly, A can verify for x_B.Therefore we say A's first message as his commitment to $\langle x_A, r_A \rangle$.

2. B can not cheat because he can not get $\langle x_A, r_A \rangle$ from first message of A and hence can not change his result.Similarly A can not cheat.
   Hence Both parties (A and B) can toss a fair random coin over the phone

1.2 Using hardcore bit for one bit encryption

Bob(B) wants to send a bit b to Alice(A).Eve(E) tries to get b. Then scheme is as follows:
1. Alice has TDP f as her public key and its trapdoor information t as her secret key.
2. Bob selects a random $x \in \{0,1\}^k$ and sends Alice cipher text $c = \langle f(x), h(x) \oplus b \rangle$.
3.Alice gets x from $f(x)$ using the trapdoor t;$h(x)$ is computed from x; b is obtained from $(h(x) \oplus b)$ using $h(x)$.

Analysis - (Security from E) - To learn anything about b, Eve must learn about $h(x)$.Here Eve only knows $f(x)$.Since, $h(x)$ is a hardcore and Eve cannot predict $h(x)$ given $f(x)$ better than flipping a coin,so b is completely secure.

2 Computational Indistinguishability

Definition 1 *Two ensembles x_k and $x_k^{'}$ are computationally indistinguishable if*
$\forall$ PPT distinguisher D,

$$Pr_{x \leftarrow x_k}[D(x) \to 1] - Pr_{x \leftarrow x_{k'}}[D(x) \to 1] \leq negl(k)$$

Informally, if given two samples to any polynomial time distinguisher D, it does not change its behavior then these samples are called computationally indistinguishable.

3 Pseudorandom Generator (PRG)

A PRG stretches a short random input to a longer output such that output still looks same.

Definition 2 *A PRG is a deterministic polynomial computational function in $G : \{0,1\}^k \to \{0,1\}^{P(k)}$ such that*

1. *$P(k) > k$*
2. *$\forall$ PPT distinguisher D,*
   $Pr_{x \leftarrow \{0,1\}^{P(k)}}(G(x) \to 1) - Pr_{y \in \{0,1\}^{P(k)}}(y \to 1) \leq negl(k)$

Theorem 1 *If f be a OWP with h be its hardcore bit then the function $G : \{0,1\}^k \to \{0,1\}^{P(k)}$ defined by $G(x) = f(x) \,||\, h(x)$ is a PRG.*

Proof. Proof By Contradiction Assume that $G(x)$ is not a PRG.
This means that $\exists$ a distinguisher C s.t.

$$Pr(C(U_{k+1}) \to 1) - Pr(C(G(x) \to 1))$$

is not negligible.
Here, U_{k+1} is Uniformly distributed string of k+1 bits.
Now we will use C to construct a PPT algorithm A that "breaks" hardcore bit h of f i.e. we will use a PPT algorithm A which computes $h(x)$ from $f(x)$ with non-negligible advantage $(\frac{1}{2} + \varepsilon)$
We construct algorithm $A(f(x) \to h(x))$ which on input $y = f(x)$,choose a random bit $b \xleftarrow{Rand.} \{0,1\}$ and run $C(y,b)$.If $(C(y,b) \to 1)$ (represent that C has identified that string is output of $G(x)$), then C outputs $h(x) = b$ else it outputs $h(x) = 1 - b$.

Clearly,$(y,b) \epsilon U_{k+1}$ because $f(x), b$ are both uniform with probability $\frac{1}{2}$.

Let $Pr(C(U_{k+1}) \to 1) = p$ then $Pr(C(G(k)) \to 1) \leq p - \varepsilon$
and

$$Pr(C(y,b) \to 1) = \frac{1}{2} * Pr(C(y, h(x)) \to 1) + \frac{1}{2} * Pr(C(y, \overline{h(x)}) \to 1) \tag{1}$$

where $\overline{h(x)}$ means $b \neq h(x)$
Thus with probability $\frac{1}{2}$ we choose $b = h(x)$ and output b with probability $< p - \varepsilon$
also with probability $\frac{1}{2}$ we choose $b = \overline{h(x)}$ and output b with probability $> 1 - (p - \varepsilon)$
i.e.

$$Pr(C(y, h(x)) \to 1) \leq p - \varepsilon \tag{2}$$
$$Pr(C(y, \overline{h(x)}) \to 1) > 1 - (p - \varepsilon) \tag{3}$$

Therefore, overall probability that A outputs $h(x)$ correctly

$$\begin{aligned} Pr(C(y,b) \to 1)) &> \frac{1}{2}(p - \varepsilon + (1 - (p - \varepsilon))) \\ &= \frac{1}{2} + \varepsilon \end{aligned} \tag{4}$$

Thus if C can break G then A can computes h(x) from f(x) with probability non-negligible $(\frac{1}{2} + \varepsilon)$. This is contradiction to the statement that h is hardcore bit of f. ■

In the next lecture we will look towards stretching of PRG outputs.