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∈ $I_n = \left\{b : \left(\frac{b}{n}\right) = +1\right\}$, for any PPT A,
$\Pr\{A(a) \to \alpha \wedge \alpha = QR(a)\} \leq \frac{1}{2} + negl(n)$.
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
   $\Longrightarrow$ g^{p-1}=1
   $\Longrightarrow$ $g^{(p-1)/2}$=$\pm$ 1
   Since g is a generator ord(g)=p-1 and hence $g^{(p-1)/2} \neq +1$
   $\Longrightarrow$ $g^{(p-1)/2}$=-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.