CS151: Intro to Cryptography and Computer Security

Midterm Exam

Andrew Chin *Due: March 21, 2007*

Problem 1: Diffie Hellman Key Exchange

a. We would like to show that Alice's key (K_A) is equal to Bob's key (K_B).

$$\begin{aligned} K_B &= K_A \\ X^y \mod p &= Y^x \mod p \\ (g^x)^y \mod p &= (g^y)^x \mod p \\ g^{xy} \mod p &= g^{yx} \mod p \end{aligned}$$

Because $xy = yx$, $K_A = K_B$.

We also would like to show that Eve is unable to compute the shared key. Here is the information that Eve knows: p, g, $g^x \mod p$, $g^y \mod p$. To compute the key, Eve would need to know both x and y, but she can not compute either x or y given the information she knows. (TODO: expand here)

b.
- When $p = 13$, $g = 2$:

x	X	
0	$2^0 \mod 13$	$= 1 = 1^2$
1	$2^1 \mod 13$	$= 2$
2	$2^2 \mod 13$	$= 4 = 2^2$
3	$2^3 \mod 13$	$= 8$
4	$2^4 \mod 13$	$= 3 = 4^2$
5	$2^5 \mod 13$	$= 6$
6	$2^6 \mod 13$	$= 12 = 5^2$
7	$2^7 \mod 13$	$= 11$
8	$2^8 \mod 13$	$= 9 = 3^2$
9	$2^9 \mod 13$	$= 5$
10	$2^10 \mod 13$	$= 10 = 6^2$
11	$2^11 \mod 13$	$= 7$
12	$2^12 \mod 13$	$= 1$

The probability that X will be a square is $\frac{6}{13}$

- The probability that Y will be a square is also $\frac{6}{13}$
- $K_A = K_B = X^y \mod p = Y^x \mod p$ will be a square $\frac{6}{13}^{th}$ of the time.
- In general, the probability that $X = g^x$ will be a square is $\frac{p-1}{2p}$. If p is a prime, and $< g >= \mathbb{Z}_p^*$, then g^x is a square iff x is even. Because $x \leftarrow \mathbb{Z}_p$ and there are $\frac{p-1}{2}$ even numbers in $\mathbb{Z}_p$, $\Pr[X \text{ is a square}] = \Pr[x \text{ is even}] = \frac{\frac{p-1}{2}}{|\mathbb{Z}_p|} = \frac{\frac{p-1}{2}}{p} = \frac{p-1}{2p}$
- The probability that $Y = g^y$ will be a square is the same probability that X will be a square

- Asking when $K_A = K_B$ will be a square is asking 'When is X^y a quadratic residue modulo p?' or:

$$\begin{aligned} & Pr[X^y = a^2 \pmod{13}] \\ = \quad & Pr[\left(b^2\right)^y = a^2 \pmod{13} | X = b^2] \\ = \quad & Pr[(b^y)^2 = a^2 \pmod{13} | X = b^2] \\ = \quad & Pr[X = b^2] \end{aligned}$$

The probability that he key K_A will be square is the same as the probability that X is square.

c.
- If $p = 23$ and $q = 11$ and $g = 4$:

i	4^i
0	1
1	4
2	16
3	18
4	3
5	12
6	2
7	8
8	9
9	13
10	6
11	1

$|\langle g \rangle| = 11 = q$
- In our example where $p = 23$, and $g = 4$, X will always be a square.
- When $p = 23$ and $g = 4$, Y will always be a square.
- When $p = 23$ and $g = 4$, X will always be a square.
- If $g \leftarrow QR_p$, then g is a square (i.e. there exists some a such that $a^2 = g \mod p$). $X = g^x$ for $x \leftarrow \mathbb{Z}_p$. Because g is a square, $X = (a^2)^x = a^{2x} = a^{x2} = (a^x)^2$. We can see that if g is a square, then X will always be a square.
- The same it true for $Y = g^y$ (because x and y are both chosen from the same distribution). $Y = (a^2)^y = a^{2y} = a^{y2} = (a^y)^2$. We can see that if g is a square, then Y will always be a square.
- $K_B = X^y \pmod{p}$. Using what we know about X and g, we see that: $K_B = X^y = (g^x)^y = g^{(xy)} = (a^2)^{(xy)} = a^{(2xy)} = (a^{(xy)})^2$. We see that K_B will always be a square if g is a square.
- In general, if $g \leftarrow QR_p$, then $K = K_A = K_B$ will always be in QR_p, which is considerably smaller than the size of $\mathbb{Z}_p$

Problem 2: ElGamal Encryption

a. We consider an encryption scheme to be secure if there exists an algorithm FakeCiphertext such that for all m, the following two distribution are equivalent:

$$D_{Enc}(m) = \{(PK, SK) \leftarrow GPK(1k); c \leftarrow Enc(PK, m); (c, m, PK)\}$$

and

$$D_{Fake}(m)\{(PK, SK) \leftarrow GPK(1k); \hat{c} \leftarrow FakeCiphertext(PK, |m|); (\hat{c}, m, PK)\}$$

b. For ElGamal encryption, here is a potential FakeCiphertext:

FakeCiphertext, on input $(PK, |m|)$ will:
Parse PK into (p, q, g, h)
Choose a random $m' \leftarrow QR_p$
Choose a random $r \leftarrow \mathbb{Z}_q$
Calculate $c' = (g^r, h^r m)$
Output c'

c. To show that this is secure, we want to show that $D_{Enc}(m) \approx D_{Fake}(m)$. Using the DDH assumption: If $D_0 \approx D_1$ then $D_{Enc}(m) \approx D_{Fake}(m)$. We can show this by showing the contrapositive, that is: If $\neg(D_{Enc}(m) \approx D_{Fake}(m))$ then $\neg(D_0 \approx D_1)$

d. Here is the reduction:

Assume we have $\mathcal{A}$ which can distinguish between D_{Enc} and D_{Fake} successfully with a probability of ϵ. $\mathcal{A}$ will take as inputs (c, m, PK) and will output a 1 if the input is from $D_{Enc}(m)$ and will output a 0 if the input is from $D_{Fake}(m)$. Using $\mathcal{A}$, we can construct a $\mathcal{B}$ which will be able to distinguish between D_0 and D_1. On input (p, g, x, y, z), $\mathcal{B}$ will output a 0 if the input is from D_0 and will output a 1 if the input is from D_1.

e. Here is how we would build $\mathcal{B}$.

On input (p, g, x, y, z):
Using p (which is a safe prime), calculate $q = \frac{p-1}{2}$
Generate $PK = (p, q, g, x)$
Generate $c = (y, m, z * m)$
$b \leftarrow \mathcal{A}(c, m, PK)$
If $b = 1$, then $\mathcal{A}$ thinks that it was given an input from D_{Enc}, so output 0. If $b = 0$, then $\mathcal{A}$ thinks that it was given an input from D_{Fake}, so output 1.

f. Assume $\mathcal{A}$'s advantage was ϵ. That is:

$$\begin{aligned} Pr[\mathcal{A} \text{ suceeds}] &= |Pr[(PK, SK) \leftarrow GPK(1^k); c \leftarrow Enc(PK, m); b \leftarrow \mathcal{A}(c, m, PK) : b = 1] - \\ &\quad Pr[(PK, SK) \leftarrow GPK(1^k), c' \leftarrow FakeCiphertext(PK, |m|); b' \leftarrow \mathcal{A}(c', m, PK) : b = 1]| \\ &= \epsilon \end{aligned}$$

... not finished ...

Problem 3: Pseudo-Random Generators and One Way Functions

a. G is a PNG iif D_{PR} is indistinguishable from D_{Random}:

$$D_{PR} = \{x \leftarrow \{0,1\}^k : G(x)\}$$
$$D_{Random} = \{R \leftarrow \{0,1\}^{2k} : R\}$$

$$|Pr[x' \leftarrow D_{PR}; b \leftarrow \mathcal{A}(x'); b = 0] - Pr[x \leftarrow D_{Random}; b \leftarrow \mathcal{A}(x); b = 0]| = \epsilon$$

b. A function $G()$ is a pseudo-random generator if there exists a probabilistic polynomial time algorithm $\mathcal{A}$ that can't distinguish between random bits and pseudorandom bits. Or: it can only distinguish between them with a probability ϵ which is negligible.

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

   $\forall$ ppt $\mathcal{A}$ $\exists$ $v()$ such that $v()$ is negligible and $Pr[x \leftarrow \{0,1\}^*; x' \leftarrow \mathcal{A}(f(x)) : f(x') = f(x)] = v(k)$

d. A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is a one-way function if there exists a negligible function $v(k)$ and a probabilistic polynomial time algorithm $\mathcal{A}$ such that if $x' = A(f(x))$, the probability that $f(x)$ equals $f(x')$ is equal to $v(k)$ (i.e., negligible)

e. A one-way function is a function that is hard to invert (that is, given $f(x)$, it is hard to figure out x. A one-way permutation is a permutation that is also hard to invert. A permutation is bijective (that is, for sets X and Y, there is exactly one $x \in X$ for every $y \in Y$ such that $f(x) = y$, while a function is not bijective (it is only surjective or injective).

f.
   - If $f()$ is a one-way function and $G()$ is a pseudo-random number generator, then $h(x) = f(G(x))$ is also one-way. The contrapositive is: If $h()$ is not one-way, then $f()$ is not one-way or $G()$ is not a pseudo-random number generator.
   - We suppose we are given an algorithm $\mathcal{A}$ that can invert $h()$, that is on input y, $\mathcal{A}$ will output x such that $h(x) = y$
   - We want to show that given such an $\mathcal{A}$, either we can build $\mathcal{B}$ which can invert $f()$ (that is, given input y, it will output x such that $y = f(x)$), or we can build $\mathcal{B}$ which can distinguish the output of $G()$ from random bits.
   - $c \leftarrow \mathcal{A}(f(x))$. If $\mathcal{A}$ succeeds, then we can build a $\mathcal{B}$ which will use $\mathcal{A}$ to obtain x' such that $f(x') = f(x)$.
     $\mathcal{B}$ will take as input x:
     $c \leftarrow \mathcal{A}(x)$
     So: $x = f(G(c))$
     If $x = f(a)$, then: $f(a) = f(G(c))$
     $a = g(c)$
     Output a
   - If $\mathcal{A}$ fails (or rather, succeeds with a negligible probability), then we can build a $\mathcal{B}$ which will use $\mathcal{A}$ to show that G is not a pseudorandom generator. That is, $\mathcal{B}$ can distinguish between ${}_DPR$ and D_{Random}.
     $\mathcal{B}$ will take as input $x = f(a)$ (where a is $2k$ random bits):
     $c \leftarrow A(x)$
     $f(G(c)) \neq x$
     $f(G(c)) \neq f(a)$
     $a \neq G(c)$
     we know that a is random, and we have just shown that a is distinguishable from $G(c)$. (note, here $\neq$ means 'distinguishable')
   - We have shown that if we assume that there exists an $\mathcal{A}$ which tries to invert h, we can construct a $\mathcal{B}$ which can either invert f, or show that the output of G is not random. Because we know that f is one-way and we know that G is pseudo-random, then our assumption that an $\mathcal{A}$ exists is false, and h must be one-way.