Crypto -HW 4

Hagai Ben Yehuda, ID num: 305237000
Jonathan Bauch, ID num: 204761233

1

2

We construct a randomized algorithm A' that operates on input $m = pq$ as follows:

1. Draw $y \in Z_m^*$ uniformly (we do that by drawing from $\{1, ..., m-1\}$ and making sure $\gcd(y, m)$ is zero, if it isn't we can factor m using y). .
2. Execute A on input $y^2 \pmod m$ and set x to be its result (note that since $y^2 \pmod m$ is a quadratic residue we will get a number and not "go catch a Stellagama stellio").
3. If $x = \pm y \pmod m$ and this step was executed less then c times (c being a constant positive integer that will affect the probability of success) go to step one if this step was executed c times, return 0.
4. Calculate $w = xy^{-1} \pmod m$.
5. Set $k = w + 1 \pmod m$
6. Set $z = \frac{k}{2}$.
7. set $q = gcd(z, m)$ and return $(q, \frac{m}{q})$

We shall now prove that A' runs in $O(t(n))$ and finds a factorization for m with probability $1 - \frac{1}{2^c}$. First note that m executes steps 1 through 3 at most c time (from the restriction in step 3) and each steps takes $O(t(n))$ steps. In addition for each execution A' passes step 3 with probability $\frac{1}{2}$, that is because y^2 has four roots in Z_m^*, and only two of them are $\pm y$, since y was chosen uniformly, the probability that the root that A returns for y^2 is $\pm y$ is $\frac{2}{4} = \frac{1}{2}$. Now we prove that if A' passes step 3 it returns a correct factorization.

From the CRT we can write $x = wy$ with

$$w = a_1(q^{-1} \pmod p)q + a_2(p^{-1} \pmod q)p$$

and $a_i \in \{\pm 1\}$ (this is because as stated in the lecture, if x is a root of y then it can be written as ly with l being a root of 1 in m), since we chose x such that $x \neq \pm y$, we know that $a_1 \neq a_2$.
Assume without loss of generality that $a_1 = 1$ and $a_2 = -1$, thus we have (w is from step 4)

$$w = (q^{-1} \pmod p)q - (p^{-1} \pmod q)p$$

Note that from fermat's little theorem we have

$$(q^{-1} \pmod p) = q^{p-2} + cp$$

$$(p^{-1} \pmod q) = p^{q-2} + rq$$

Thus

$$w = q^{p-1} - p^{q-1} \pmod{pq}$$

Note that

$$q^{p-1} + p^{q-1} = 1 \pmod{p}$$

and

$$q^{p-1} + p^{q-1} = 1 \pmod{q}$$

Hence

$$q^{p-1} + p^{q-1} \pmod{pq}$$

Thus

$$w + 1 = q^{p-1} - p^{q-1} + 1 = q^{p-1} - p^{q-1} + q^{p-1} + p^{q-1} = 2q^{p-1} \pmod{pq}$$

Therefore when we calculate z in step 6 we obtain q^{p-1} and obviously $\gcd(q^{p-1}, pq) = q$, and thus we indeed recover q and p in step 7 as required, since we got to step 4 in $O(t(n))$ steps and 4 through 7 also take $O(t(n))$ steps, A' is an algorithm as request. Randomization is required in our algorithm as we must get a root that is differs from the root we know not only by sign. Since we have no knowledge of what root A will return, and since we cant find another root by ourselves, we must hope A returns a different root, by choosing y randomly many times, the probability we will indeed find a root that is different not only by sign approaches 1.

3

4

5

We construct a polytime algorithm A' that on input p, g, g^x does the following:

- Draw $x \in \mathbb{Z}^*{}_p$ uniformly.
- Execute A on p, g, g^{x+y} (note that $g^{x+y} = g^x g^y$), set z to be the result.
- If $g^z = g^{x+y}$ return $z - y$, else if this is the 700'th time return 0, else go to the first step.

First note that this algorithm is polynomial as it executes A at most 700 times, and A is polynomial. For each iteration the probability of landing within the subset of $x's$ for which A finds and inverse is $\frac{1}{1000}$ as the sum of a uniform random variable and a constant is uniform. Hence with probability $\frac{1}{1000}$ we obtain the correct z in the last step, note that

$$g^{z-y} = g^z g^{-y} = g^{x+y} g^{-y} = g^x$$

Thus $z - y$ is a solution to the DL problem. The last step in A fails only if $x + y$ is not inside the set for which A solves the DL problem, this probability is $\frac{1}{1000}$ because $x + y$ distributes uniformly over $\mathbb{Z}^*{}_p$.

Because A' makes 700 tries before returning with a false result, the probability that A' fails is the probability that A fails at each attempt which is $(1 - \frac{1}{1000})^{700} < \frac{1}{2}$. Thus A' is an algorithm as requested.

6

6.a

We construct a the decryption function:

$$Dec(c_1, c_2) = \begin{cases} 1 & \text{if } c_1^x = c_2 \\ 0 & \text{else} \end{cases}$$

If $b = 0$, then $c_2 = h^y = g^{xy} = c_1^x$, thus if $b = 0$ Dec returns the correct result.
If $b = 1$ then given z there is exactly one value of y for which $g^y = g^{zx}$ since g is a multiplicative generator, if $g^y = g^{zx}$ then $y = zx \pmod{p-1}$, the only case in which we decrypt a 1 to 0 is if $y = xz$ which happens with probability at most $\frac{2}{p-1}$. Thus correct and efficient decryption is possible except for a negligible probability.

6.b

Assume that this encryption scheme is not ϵCPA secure, then there is a polynomial adversary A that wins the adversarial indistinguishability test with probability $> \frac{1}{2} + \epsilon$. We construct a polynomial time adversary A' that shows DDH is not hard: Given input (g^x, g^y, g^z) our algorithm does the following:

- Supply A with (p, g, g^x).
- Get the two messages from A assume WLOG A replays with $m_0 = 0$, $m_1 = 1$ (if this is not the case we can construct an algorithm B that is based on A and wins with the same probability, since if the messages are in a different order B can change the order and if both messages have the same value A can only guess which bit was chosen as both will be encrypted to the same value and B can supply us with two messages and also guess and win with the same probability).
- Supply A with (g^y, g^z), if A returns 1 return 0, else return 1.

We shall now show that A' distinguishes (g^x, g^y, g^z) from (g^x, g^y, g^{xy}):

$$\begin{aligned}
&\Pr_{x,y \leftarrow U_{\mathbb{Z}^*p}, z=xy}(A'(g^x, g^y, g^z) = 1) - \Pr_{x,y,z \leftarrow U_{\mathbb{Z}^*p}}(A'(g^x, g^y, g^z) = 1) \\
&= \Pr(A'(g^x, g^y, g^z) = 1 | x, y \leftarrow U_{\mathbb{Z}^*p}, z = xy) - \Pr(A'(g^x, g^y, g^z) = 1 | x, y, z \leftarrow U_{\mathbb{Z}^*p}) \\
&= \Pr(A \text{ wins } | b = 1) - \Pr(A \text{ loses } | b = 0) \\
&= 2[\Pr(A \text{ wins } \cap b = 1) - \Pr(A \text{ loses } \cap b = 0)] \\
&= 2[\Pr(A \text{ wins } \cap b = 1) - \Pr(b = 0) + \Pr(A \text{ wins } \cap b = 0)] \\
&= 2[\Pr(A \text{ wins }) - \Pr(b = 0)] \\
&\geq 2[\frac{1}{2} + \epsilon - \frac{1}{2}] = 2\epsilon
\end{aligned}$$

Note that in our calculation we refer to the probability that x, y, z are drawn uniformly or $z = xy$ (this is b as defined in the adversarial indistinguishability test) , each case has probability $\frac{1}{2}$ as we are in a distinguisher setup and thus are supplied with a sample from each distribution with equal probability (otherwise the streams are distinguishable by always saying that the current input originated from the stream with higher probability to be sampled). Thus A' is a distinguisher as required.