HomeWork5

Problem 1：

我们首先容易得到$N=35=5*7$所以说我们可以计算 N 的 Euler phi 函数值及：$\phi(N)=(5-1)^1*(7-1)^1=24$而我们根据欧拉定理可以得到$\forall x\in(\mathbb{Z}_N)^*, x^{\phi(N)}=1\ in\ \mathbb{Z}_N$我们将$101^{4,800,000,023}=(2*35+31)^{4,800,000,023}$容易得到原式子转换为$31^{4,800,000,023} mod 35$然后利用欧拉定理得到将原式子化简得$31^{23} mod 35=31^{-1} mod 35$。

而求 mod 下的逆元，我们可以用扩展欧几里得算法求$a*31+b*35=1$中的 1 值，计算得 a=26，所以原式子的值为 26。

Problem 2：

我们假设使用欧拉函数得到的值为x_1，及$\phi(N)=p^{1-1}(p-1)*q^{1-1}(q-1)=x_1$，我们有$N=p*q$那么我们可以得到与 p 相关的式子：$p^2-(N+1-x_1)p+N=0$。接着我们只要证明对这个式子的求解得到的 p 是多项式时间的即可，我们利用求根公式得到：

$$p=\frac{(N+1-x_1)\pm\sqrt{(N+1-x_1)^2-4N}}{2}. \tag{1.1}$$

接下来对计算该式子的时间进行分析：首先加减法均是线性时间可以计算出来的，其次乘法与除法是多项式时间可以计算的得到的。对于开方运算，我们知道这个结果中求根所得到的结果是有理数，所以我们可以用直接用二分法对其进行计算，可以得到其时间的上界为：$O(\log(N^2)N^2)$，其也是多项式时间可以计算得到的，实际计算的时间经过改进肯定远小于此时间。

因此我们可以得到由于计算中涉及到的所有计算都是多项式时间内可以完成的，所以求出 p 的时间是多项式的，所以求出 q 的时间也是多项式的，因此得证。

Problem 3：

首先介绍下算法的基本思想，首先由于 N 为两个大质数的乘积，所以 2 这个元素一定属于$\mathbb{Z}_N{}^*$，所以根据扩展欧几里得算法可以在多项式时间求得$y=2^{-1} mod N$。

之后我们可以分析$x^e mod N$的组成。首先如果其最低位为 LSB（x），**假设x去掉最低位后得到的值为x'**，则我们可以得到$x^e mod N=(x'*2+LSB(x))^e mod N$。所以我们

在$x^e modN$的值上乘y^e之后 mod N 可以得到:

$$((x^e modN)\cdot y^e)modN=(x^e\cdot y^e)modN=(x'+LSB(x)\cdot y)^e modN \tag{1.2}$$

我们之后的计算可以在这个的基础上进行。当 LSB（x）为 0 时我们直接在$x^e modN$的值上乘y^e可以得到$(x')^e modN$，接着继续进行即可。要是 LSB(x)不为 0，我们得到的值为：$(x'+y)^e modN$，此时没法直接得到$(x')^e modN$，因此无法直接调用题中的函数，我们采用下面的方法:

x 的最右 l 比特与$2x''$-N 的最右 l 比特相同，其中x''是通过$(x\cdot y)modN$计算得到的最右 l-1 比特，这样就得到了长度有关的递归式，下面证明这个结论是对的:

$$\begin{aligned}((2*(x*y)\ mod\ N)\ mod\ 2^{l-1}-N)mod\ 2^l&=((x*y*2)mod\ N-N)\ mod\ 2^l\\&=(x\ *y*2)\ mod\ 2^l\\&=x\ mod\ 2^l\end{aligned}$$

所以我们可以通过递归反复的进行$(x^e\cdot y^e)modN$操作来得到我们想要的最后一位。

Algorithm 1

INPUT N，e，c，l（其中 l 需要计算得到的 x 的位数，从后往前）
Output C 对应的明文
If l = 1 **then**

return $\mathcal{A}(N,c)$

Else

$x_1=A(N,c)$，$y=2^{-1}\ mod\ N$

$x'=$ Algorithm(N，e，$(c\cdot y^e)mod\ N$，l-1)

if $x_1=0$ **then**

return $x'||x_1$

else

return $(2x'-N)\ mod\ 2^l$

end
end

可以发现我们把当 c 的最后一位为 1 时无法计算的情况转换为了算$(c\cdot y^e)mod\ N$的 l-1

位，由于递归到 l 为 1 时我们是可以直接计算的，所以可以通过递归计算，递归到递归终止条件后再从后往前计算$(2x'-N)\ mod\ 2^l$即可。

Problem 4:

首先我们说明两人共享相同的密钥，即 Bob 最终收到的密文是什么：

$$w \oplus t = u \oplus r \oplus t = s \oplus r = k$$

因此 Bob 最终收到的密钥与 Alice 的密钥相同。

但是，这个密钥交换方法Π是不安全的，我们用安全密钥交换实验$KE^{eva}_{\mathcal{A},\Pi}$来分析：构造如下的攻击者$\mathcal{A}$，其利用 Alice 与 Bob 的对话记录 trans 来进行攻击。其首先将对话记录中的所有消息异或，得到了：$s \oplus u \oplus w = k$，因此我们通过二者的对话记录就可以直接得到密钥，所以我们的实验成功的概率为：

$$Pr(KE^{eva}_{\mathcal{A},\Pi}(n)=1)=1-\frac{1}{2}Pr\left(\hat{k}=k\right)=1-\frac{1}{2^{n+1}}>\frac{1}{2}+negl(n)$$

所以该密钥交换方案为不安全的。

Problem 5：

（a）其解密方法可以通过计算$c_1{}^x$看它是否与c_2相等，如果相等则输出 0，否则输出 1。由于其解密有一定概率可能出错，我们验证一下其是否符合公钥加密的定义：计算得$Pr(Dec_{sk}(Enc_{pk}(b))=b)=1-\frac{1}{2^n}$所以其符合公钥加密得定义。

（b）我们将加密方法Π的安全性归约到 DDH 难问题上：具体的我们假设攻击者$\mathcal{A}$来攻击加密方案Π，然后我们构造攻击者$\mathcal{A}'$来攻击 DDH 问题。具体的当进行实验时$\mathcal{A}'$收到$(\mathbb{G},q,g,g^x,g^y,g^w)$其中$w$可以是$g^{xy}$（意味着 b=0）也可以是$g^z$（意味着 b=1）其中 y 与 z 满足：$y,z\xleftarrow[random]{}\mathbb{Z}_q$。且$\mathcal{A}'$平常可以自由的向数据库询问任何一种情况，使其攻击对象产生一组新的$(\mathbb{G},q,g,g^x,g^y,g^w)$值，其中 y 与 z 每次重新生成，**均为随机数**。

接着$\mathcal{A}'$利用收到的消息中的(G,q,g,g^x)作为公钥，攻击者$\mathcal{A}$可以向$\mathcal{A}'$询问，当其想加密 1 时$\mathcal{A}'$向数据库询问后输出$<g^y,g^z>$，加密 0 时$\mathcal{A}'$向数据库询问后输出$<g^y,g^{xy}>$。需要注意的是由于每次$\mathcal{A}'$查询时产生的 y 与 z **均为随机的**，因此引入了随机性使得 cpa 攻击

失效。

加密时$\mathcal{A}'$将输入的消息$<g^y, g^w>$发送给攻击者$\mathcal{A}$。并且当$\mathcal{A}$输出 0 时$\mathcal{A}'$输出 0 ，否则$\mathcal{A}'$输出 1。那么我们有：

$$\begin{aligned}|2 \cdot Pr(PubK^{cpa}_{\mathcal{A},\Pi}(n)=1)| &= |Pr(PubK^{cpa}_{\mathcal{A},\Pi}(n=1)=1\,|b=0) + Pr(PubK^{cpa}_{\mathcal{A},\Pi}(n=1)=1\,|b=1)| \\ &= |1 - Pr(\mathcal{A}'(\mathbb{G},q,g,g^x,g^y,g^{xy})=1) + Pr(\mathcal{A}'(\mathbb{G},q,g,g^x,g^y,g^z)=1)| \\ &= 1 + w(n)\end{aligned}$$

由于 DH 问题是难解的所以有$w(n)$为可忽略的，所以我们有：

$$Pr(PubK^{cpa}_{\mathcal{A},\Pi}(n)=1) = \frac{1+w(n)}{2} \leq \frac{1}{2} + negl(n) \tag{1.3}$$

所以原加密方案为 CPA 安全的。

■

Problem 6：

由于 El Gamal 加密方案中我们如果知道g^r以及公钥$(\mathbb{G},q,g,h)$是无法得到h^r的，因为攻击者无法由g^r求出 r 或者由h求出 x，因此无法用乘方的方法直接求h^r或者是用$g^{r\cdot x}$的方法来求。

因此我们可以直接把h^r作为私钥加密方案的**密钥**，类似的我们发送消息：$<g^r, Enc_{h^r}(m)>, r \xleftarrow[random]{} \mathbb{Z}_q$。而攻击者无法得到密钥$h^r$自然无法对其进行有效攻击，当接收方收到后只要用公钥对应的私钥 x 做$g^{r\cdot x}$的操作即可得到私钥加密的密钥h^r。

综上 $Enc^{hy}: r \leftarrow \mathbb{Z}_q, k \leftarrow h^r, c \leftarrow Enc_k(m)$； $Dec^{hy}: k := Dec_{sk}(h^r) = h^{r\cdot x}, m := Dec'_k(c)$。

Problem 7：

(a) 我认为它是不安全的我们可以选择$m' = m^{-1}\ mod\ N$作为我们对 signing 数据库的输入，我们计算可得$\sigma' = (m^{-1})^d = (m)^{-d}\ mod\ N$，那么因此我们可以伪造原消息的 signature：

$$(\sigma')^{-1} mod\ N\ =\ (m)^d\ mod\ N = \sigma$$

所以这个签名方案不是安全的。接下来分析这么做的正确性：首先$m \in \mathbb{Z}_n{}^*$，这样才能用扩展欧几里得算法求逆，而这是 RSA 的前提，是满足的，同样的由于σ'也属于$\mathbb{Z}_N{}^*$，所以σ'

也是可以求逆的。所以这种做法是可行的，因此此加密方案是不安全的。

(b) 我认为它是安全的。如果不是的话我们假设存在一个攻击者，其可以只根据公钥 e 与 m 就可以得出m^d，那么我们可以知道任何的 RSA 加密方案都可以用它来破解，我们只需要输入公钥 e 以及$c = m^e$它会自动得到$(m^e)^d = m$，而这显然是不可能的。因此这个加密方案是安全的。

Problem 8：

非常容易地我们可以想到只要满足$m_0 \oplus m_1 = 1^l$即可，例如$m_0 = 0^l, m_1 = 1^l$，我们因此得到$\sigma_0 = (x_{1,0}, x_{2,0}, \cdots, x_{l,0}), \sigma_1 = (x_{1,1}, x_{2,1}, \cdots, x_{l,1})$我们可以很容易的构造加密矩阵，见老师上课的 ppt 如下：

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}$$

所以现在我们可以加密伪造任何消息了，攻击完毕。