Assignment: Due Monday, 11:59pm

Chris Betsill

October 17, 2016

1. Diffie-Hellman Intruder-in-the-Middle attack.

Suppose Alice and Bob conduct a Diffie-Hellman key exchange as follows.

1. Alice and Bob agree on prime $p = 240922393$ and primitive root $\alpha = 103$.
2. Alice chooses secret $x = 120461539$ and sends α^x to Bob.
3. Bob chooses secret $y = 81635591$ and sends α^y to Alice.

Compute the shared secret key. Remember to do your calculations in $U(p)$.

```
p <- 240922393
alpha <- 103
x <- 120461539
y <- 81635591

k <- powm(powm(alpha, x, p), y, p)
print(k)
```

```
Big Integer ('bigz') :
[1] 115073581
```

Suppose now that Eve has been able to intercept messages between Alice and Bob and replace them with her own messages. In the above transaction, Eve secretly does the following.

- Eve chooses a "fake" exponent $z = 49182605$.
- Eve intercepts Alice's message α^x and sends Bob α^z instead.
- Eve intercepts Bob's message α^y and sends Alice α^z instead.

Compute the keys that Alice and Bob would compute under this scenario. Call these keys `keyA` and `keyB`.

```
p <- 240922393
alpha <- 103
x <- 120461539
y <- 81635591
z <- 49182605

keyA <- powm(powm(alpha, x, p), z, p)
keyB <- powm(powm(alpha, y, p), z, p)

print(keyA)
```

```
Big Integer ('bigz') :
[1] 131722659
```

```
print(keyB)
```

```
Big Integer ('bigz') :
[1] 97466269
```

▷ Explain why Eve is also able to compute `keyA` and `keyB`.

Eve was able to intercept both α^x and α^y. Since she generated z, it is trivial to compute $(\alpha^x)^z$ and $(\alpha^y)^z$. The modulus p, was public.

▷ Suppose that Alice and Bob try to use a symmetric cipher to communicate using `keyA` and `keyB`. Explain, in your own words, how Eve could listen in on their conversation without being detected. (See page 258 if you are stuck.)

Since she has both keys, she can intercept messages, decrypt them with the corresponding key, and rencrypt with the opposite key, and send them to the intended recipent with neither Alice or Bob noticing.

2. ElGamal example

Alice and Bob are communicating using the ElGamal cryptosystem with $p = 179841529021446883498969891$ and $\alpha = 7$. Bob's secret exponent is $a = 10000$.

Give the triple that Bob makes public.

```
alpha <- 7
p <-  as.bigz("179841529021446883498969891")
a <- 10000

beta <- powm(alpha,a,p)

cat(sprintf("(%f, %f, %f)", p, alpha, beta))
```

```
(179841529021446881271611392.000000, 7.000000, 562436939196737721865338888.000000)
```

Using this public key, Alice sends the ciphertext pair

$$(129572415000137218728447332, 172765025692695167473988064)$$

to Bob. Decrypt. Use your `bigzToString` function to convert your answer to a string.

```
library(cryptoCTB)
r <- as.bigz("129572415000137218728447332")
t <- as.bigz("172765025692695167473988064")
p <-  as.bigz("179841529021446883498969891")
a <- as.bigz("10000")
ra <- powm(r, a, p)
ira <- inv.bigz(ra, p)
m <- (t*ira) %% p
print(bigzToString(m))
```

```
[1] "Javier Baez"
```

3. Compute a discrete log modulo 2.

In $U(534576346787345678345673678673460 03)$, compute

$$L_5(312241123030639599198802886791256 45)$$

modulo 2. Show your work.

```
p <- as.bigz("534576346787345678345673678673460 03")
beta <- as.bigz("312241123030639599198802886791256 45")
twoc <- as.bigz((p-1))
c <- twoc/2
x <- powm(beta, c, p)
x
```

```
Big Integer ('bigz') :
[1] 534576346787345678345673678673460 02
```

4. Apply the Chinese Remainder Theorem

Use your `crtSolve` function to compute $L_3(47473199372)$ modulo 198123889721. Use the following facts.

$$\begin{aligned} L_3(47473199372) &\equiv 1 \pmod 5 \\ L_3(47473199372) &\equiv 7 \pmod 8 \\ L_3(47473199372) &\equiv 37 \pmod{71} \\ L_3(47473199372) &\equiv 1977 \pmod{2371} \\ L_3(47473199372) &\equiv 11395 \pmod{29423} \end{aligned}$$

```
print(crtSolve(c(1,7,37, 1977, 11395), c(5,8,71,2371,29423)))
```

```
Big Integer ('bigz') :
[1] 15240301551
```

5. Test Brute Force times vs. Pohlig-Hellman times

1. Use the `nextprime` and `nextPrimRoot` functions to find a prime p, a primitive root α, and a power β of α such that your `discreteLogBrute` function takes more than 2 seconds (but less than a minute) to find the discrete logarithm $L_\alpha(\beta)$ in $U(p)$.

```
p <- 3156001
alpha <- nextPrimRoot(p)
alpha
```

```
[1] 31
```

```
beta <- 137
print(system.time(discreteLogBrute(alpha, beta, p))[3])
```

```
elapsed
 11.206
```

2. Use the `nextprime` and `nextPrimRoot` functions to find a prime p, a primitive root α, and a power β of α such that *this online implementation of the Pohlig-Hellman algorithm* takes more than 2 seconds (but less than a minute) to find the discrete logarithm $L_\alpha(\beta)$ in $U(p)$.

```
beta <- 17
p <- 13107181911273173
alpha <- nextPrimRoot(p)
alpha
```

```
[1] 2
```