Physics 4481-7681; CS 4912 Problem Set 4

Alex Hahn March 27, 2014

Problem 1: Defeating RSA encryption with period finding. I
Here we examine RSA encryption and how it can be defeated by an efficient period-finding program, by working everything out in a particular case. To get a better feeling for what is involved, try doing this without a computer or calculator.

(a) The two numbers Bob announces publicly are $N = 55$ and $c = 17$. Let Alice's message a be 9. What number between 1 and 54 is Alice's encrypted message $b \equiv a^c (\text{mod } 55)$? Rather than using a calculator, note that 17 in binary is 10001, so you can work it out efficiently by listing 9^2 mod 55, 9^4 mod 55 etc. Your write-up should list the values of all the powers of 9 modulo 55 that you had to calculate to construct b. (If easier, a few can be given as negative numbers modulo 55.)

$$9 \equiv 9(\text{mod } 55)$$
$$9^2 \equiv 81 \equiv 26(\text{mod } 55)$$
$$9^4 \equiv (9^2)^2 \equiv 26^2 \equiv 16(\text{mod } 55)$$
$$9^8 \equiv (9^4)^2 \equiv 16^2 \equiv 36(\text{mod } 55)$$
$$9^{16} \equiv (9^8)^2 \equiv 36^2 \equiv 31(\text{mod } 55)$$

Since we are looking for 9^{17} we can use modular arithmetic to show

$$9^{17} = 9^1 \cdot 9^{16} \equiv 9 \cdot 31 \equiv 279 \equiv 4(\text{mod } 55)$$

so 4 is the encrypted message.

(b) Since you have in your head the computational resources needed to find the prime factors of 44, you are in a position to find Bob's decoding number d. What is it? (This can be done using the Euclidean algorithm described in Appendix I.) Confirm that $b^d \equiv a(\text{mod } 55)$, by the same process of successively squaring used in (a).

Using Euclid's algorithm we have

$$40 = 17 \cdot 2 + 6$$
$$17 = 6 \cdot 2 + 5$$
$$6 = 5 \cdot 1 + 1 \rightarrow -2(\text{mod } 40)$$
$$5 = 1 \cdot 5 + 0 \rightarrow 5(\text{mod } 40)$$
$$\rightarrow 33(\text{mod } 40)$$

so 33 is our answer for d the decoding number. Now we check that $4^{33} \equiv 9(\text{mod } 55)$

$$4 \equiv 4(\text{mod } 55)$$
$$4^2 \equiv 16(\text{mod } 55)$$
$$4^4 \equiv (4^2)^2 \equiv 16^2 \equiv 36(\text{mod } 55)$$
$$4^8 \equiv (4^4)^2 \equiv 36^2 \equiv 31(\text{mod } 55)$$
$$4^{16} \equiv (4^8)^2 \equiv 31^2 \equiv 26(\text{mod } 55)$$

$$4^{32} \equiv (4^{16})^2 \equiv 26^2 \equiv 16 (\text{mod } 55)$$

from here we can see that

$$4^{33} = 4^1 \cdot 4^{32} \equiv 4 \cdot 16 \equiv 9 \text{mod } 55 \checkmark$$

(c) Eve, listening in to the public communications picks up Bob's publicly announced $N = 55$ and $c = 17$ as well as Alice's encoded message b. Using her quantum computer she calculates the period r of b modulo N. What is r? Although you lack a quantum computer you can factor N in your head, and therefore know the order of G_N. Since r must divide that order there are not very many possibilities to examine.

r must be one of the factors of 40 because $G_N = 4 \cdot 10 = 40$. after checking a few of them we find that

$$4^{10} \equiv 1 (\text{mod } 55)$$

$$\therefore \ r = 10$$

(d) Find the inverse d' of c modulo r. (This is simple enough not to need Euclid's algorithm.) Confirm that $b^{d'} \equiv a (\text{mod } 55)$

$$17d' \equiv 1 (\text{mod } 10)$$

$$7d' \equiv 1 (\text{mod } 10)$$

$$7d' \equiv 21 (\text{mod } 10)$$

$$d' \equiv 3 (\text{mod } 10)$$

confirming $b^{d'} \equiv a (\text{mod } 55)$:

$$4^3 \equiv 64 (\text{mod } 55)$$

$$4^3 \equiv 9 (\text{mod } 55) \checkmark$$

Problem 2: Defeating RSA encryption with period finding. II
Now suppose Bob sends $N = 143$ and $x = 53$ over a public channel to Alice, who uses them to encode her message a and sends back the encoded message $b = 19$ to Bob. (In the below, feel free to use a classical computer or classical pocket calculator if useful.)

(a) Eve uses her quantum computer to determine that the period of the function $f(x) \equiv 19^x (\text{mod } 143)$ is $r = 60$. Find d' (using, e.g., Euclid's algorithm) such that $cd' \equiv 1(\ \text{mod} r)$, and use that to recover Alice's original message a.

Using Euclid's algorithm:

$$60 = 53 \cdot 1 + 7$$

$$53 = 7 \cdot 7 + 4$$

$$7 = 4 \cdot 1 + 3 \rightarrow -1 \text{mod } 40$$

$$4 = 3 \cdot 1 + 1 \rightarrow 8 \text{mod } 40$$

$$3 = 1 \cdot 3 + 0 \rightarrow -9 \text{mod } 40$$

$$\rightarrow 17 \text{mod } 40$$

therefore $d' = 17$ and recovering Alice's original message we have $19^{17} \equiv 2 (\text{mod } 143)$ so $a = 2$
(b) Bob knows instead the prime factors p, q of N. Use those together with Euclid's algorithm to determine d such that $cd \equiv 1 (\text{mod } (p-1)(q-1))$, then use d to decrypt Alice's original message a, and see if Eve

got it right.

Using Euclid's algorithm

$$120 = 53 \cdot 2 + 14$$
$$53 = 14 \cdot 3 + 11$$
$$14 = 11 \cdot 1 + 4 \rightarrow -2$$
$$11 = 3 \cdot 3 + 2 \rightarrow 7$$
$$3 = 2 \cdot 1 + 1 \rightarrow -9$$
$$2 = 1 \cdot 2 + 0 \rightarrow 34$$
$$\rightarrow -43$$

so d, the decrypting code is -43. Checking $19^{-43} \equiv 2 (\text{mod } 143)$ so Eve was correct.

Problem 3: Discrete Fourier Transform
In class (lecture 14, and noted linked from course website,) we emulated the steps to factor the number $N = 15$ via period finding. Here we will consider two problems related to factoring $N = 21$:

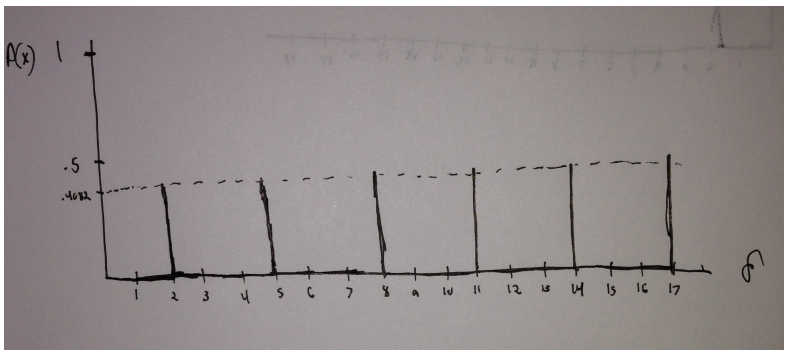
(a) First we'll consider the function $f(x) = b^x \text{mod } 21$, with $b = 4$. For experience with discrete Fourier transform, imagine starting from a state $|\Phi\rangle = \frac{1}{\sqrt{18}} \sum_{x=0}^{17} |x\rangle |f(x)\rangle$ (though note this state can't be easily arranged with Hadamard gates in a conventional Qbit quantum computer, since 18 is not a power of 2).

(i) Suppose we measure the "output" $|f(x)\rangle$ as 16. In what state $|\Psi\rangle$ will that leave the "input"? Writing the state in the form $|\Psi\rangle = \sum_{x=0}^{17} \gamma(x) |x\rangle$, graph the (discrete) function $\gamma(x)$ for values from 0 to 17.

The input is left in the state

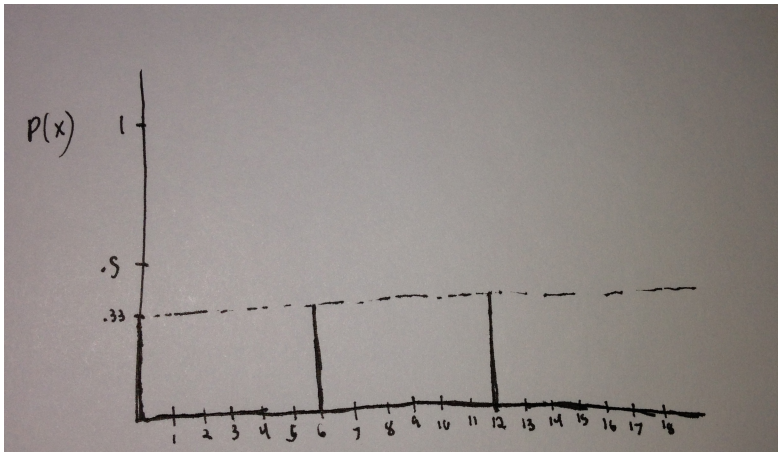
$$|\Psi\rangle = \frac{1}{\sqrt{6}}(|2\rangle + |5\rangle + |8\rangle + |11\rangle + |14\rangle + |17\rangle)$$

We can visualize this in the frequency domain as a series of delta functions with amplitude $\frac{1}{\sqrt{6}} \approx 4.0824$ at the corresponding states



(ii) Now calculate the discrete Fourier transform $\tilde{\gamma}(y) = \frac{1}{\sqrt{18}} \sum_{x=0}^{17} e^{2\pi i x y/18} \gamma(x)$ (as given in eq 3.20 of course text), and graph the values of the modulus squared $|\tilde{\gamma}(y)|^2$ for y from 0 to 17.

writing a program in Matlab to calculate the sum over the terms I get an answer of
`1.93e-15(Re)+8.37e-16(Im)` with the corresponding graph:



(iii) Imagine now measuring the state $U_{FT}\,|\Psi\rangle = \sum_{x=0}^{17} \tilde{\gamma}(y)\,|y\rangle$. With what probability would the various non-zero values of y be measured?

The non-zero values of y each have probability $\frac{1}{3}$.

(iv) Unlike a classical discrete Fourier transform, in the quantum case the measurement in part (iii) only reveals a single Fourier transform component. Presuming you didn't already know the value of the period r of the above $f(x)$, how could you infer the period from any one of those possible measured values of y?

(section 3.7)

So we run into a question of finding the period r in a one computation "take all" scheme. The quantum Fourier transform provides us with a unitary transformation that takes the x_0 dependence into a "harmless" overall phase factor. Now to find r we note that the probability function

$$p(y) = \frac{1}{2^n m}\left|\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n}\right|^2$$

is a function of the integer y whose magnitude has maxima when y is close to integral multiples of $2^n/r$. Assuming that we have taken enough input registers ie 2^n is as large as N^2 so it is large enough to contain at least N full periods we now look at equation 3.57

$$\left|\frac{y}{2^n} - \frac{j}{r}\right| \leq \frac{1}{2^{n+1}}$$

Here we know n and y is the result of our measurement so we have an estimate for j/r. BUT because we have used twice as many Qbits as needed to represent all the integers up to N we have ensured that our estimate j/r is off by no more than $1/(2N^2)$. From here we can figure out j/r (with our knowledge of $y(measured)/18$) with the method of continued fractions (detailed in appendix K). Note that the method gives us j_0 and r_0 with no common factors such that $j_0/r_0 = j/r$. We then use the fact (result from Appendix J) that two random numbers j and r have a fairly decent chance of having no common factors. We check to see whether r_0 is infact r by seeing if $b^{r_0} (\text{mod } N) = b$. If this is not the case we can smaller low multiples (it is rare that two random numbers share large factors). If even this doesn't work out that great the book details how we can repeat the measurement procedure and check the math to see if we found r and in fact if we have enough additional Qbits we have a very high chance of finding r.

(b) (i) there are 10 such values, $|\Psi\rangle$ (the input state) looks like

$$|\Psi\rangle = \frac{1}{\sqrt{10}}(|4\rangle + |10\rangle + |16\rangle + |22\rangle + |28\rangle + |34\rangle + |40\rangle + |46\rangle + |52\rangle + |58\rangle)$$

(ii) Note first off that m is 10 and $n = 5$ for us in the equation (from book)

$$p(y) = \frac{1}{2^n m} \frac{\sin^2(\pi \delta m r/2^n)}{\sin^2(\pi \delta r/2^n)}$$

plugging in our values we indeed get

$$p(y) = \frac{1}{2^5 \cdot 10} \frac{\sin^2(\pi 6 \cdot 10y/2^5)}{\sin^2(\pi 6y/2^5)}$$
$$= \frac{1}{640} \frac{\sin^2(\pi 60y/64)}{\sin^2(\pi 6y/64)}$$

Using MATLABS handy anonymous function capability

```
f=@(y)(1/640)*(((sin((pi*60*y)/64))^2)/((sin((pi*6*y)/64))^2))
```

we can quickly find the 5 most likely y value probabilities (just iterate through 0 to 64, storing in an array, and sort)

```
arr=[];
for i= 1:63 %y=0 returns NaN
    arr(i)=f(i);
end

sarr=sort(arr);
```

I find that the top 5 probabilities are all .1124 (I actually get that one value is 1.3451? also the sum of all entries is 2.0326, which is > 1, are we supposed to normalize?). If we take the top 5 (including the 1.3451) we get a sum of 1.7947 (this is pretty high out of a total of 2 so it's a good chance we return one of these) (without it we get about .6)

(iii) We follow the same logic/ procedure as explained in 3(a iv) but now with $y/64$. or you could try to reverse solve

$$p(y_i) = \frac{1}{r} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j} \right)^2$$

(since we have the probabilities ?)

Problem 4: Factoring a product $(2^k + 1)(2^l + 1)$

Suppose we are asked to factor the number $N = 16843009$, which is somehow known in advance to be the product of two primes of the form $p = 2^k + 1$, and $q = 2^l + 1$. This is the special case mentioned on pp. 81-82 of the course text, in which the period is necessarily a power of 2 (since it divides $(p-1)(q-1)$). The smallest n_0 such that $2^{n_0} > N$ in this case is $n_0 = 25$, and as explained in the texr we can take the number of Qbits to be $n = n_0$ (rather than $2n_0$). Feel free to uses a classical computing aid for the arithmetic below, and although N is small enough to factor directly by classical methods, try instead to follow the steps of the quantum argument.

(a) Following the method in section 3.10, suppose we pick $a = 255$ (coprime to N), and we wish to find the period r of the function $a^x \text{mod } N$ using our $n = 25$ bit quantum Fourier transformer. Find the total

probability of measuring y to be an integer multiple of $2^n/r$, verify the argument on p 81 that this probability is "essentially" unity, With what probability does measuring y give no useful information to determine r?

From the book we have

$$p(y_j) = \frac{1}{2^n m}\left(\frac{\sin(\pi\delta_j)}{\pi\delta_j r/2^n}\right)^2 = \frac{1}{r}\left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2$$

We also know that if y_j is an multiple of $2^n/r$ then $\delta_j = 0$ and that there are $r-1$ terms in this summation. Lastly we will use the fact that r is large. Our summation boils down to:

$$\sum_{j=1}^{r-1} p(y_j) = \frac{1}{2^n m}\left(\frac{\sin(\pi\delta_j)}{\pi\delta_j r/2^n}\right)^2 = \frac{1}{r}\left(\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right)^2 = \frac{1}{r} \approx 1 \checkmark$$

(b) Now suppose we measure $y = 7 \cdot 2^1 9$. This determines j/r directly without having to use continued fractions, and gives r up to any common divisors with j. Verify that the r you obtained does indeed satisfy $a^r \bmod N \equiv 1$.

rearranging we have

$$\begin{aligned} r &= \frac{7 \cdot 2^{25}}{2^{19}} \\ &= 448 \end{aligned}$$

checking in wolfram alpha 255^{448} mod 16843009 does equal 1 ✓

(c) Evaluate $t = a^{r/2} \bmod N$, and use it to determine the prime factors of N by the method in section 3.10, by finding the greatest common divisors of N with $t+1$ and $t-1$ ($a = 255$ has already been chosen so that $t + 1 \neq 0 \bmod N$, as specified in eq (3.67)).

$$t = a^{r/2} \bmod N = 65536 = 2^{16}$$

$$p = (N, t-1) = 257$$

$$q = (N, t+1) = 2^{16} + 1 = 65537$$

Problem 5: Period Finding and continued fractions

This illustrates the mathematics of the final (post-quantum-computational) stage of Shor's period finding procedure, as described on p.82 and appendix K of the course text. Make use of the theorem, cited in the text, that if j and r have no common factors, and x is an estimate for the fraction j/r that differs from it by less than $1/2r^2$, then j/r will appear as one of the partial sums in the continues fraction expansion of x

(a) Suppose you know that the integer r is less than 100 and that 7080 is within $\frac{1}{2}$ of an integral multiple of $2^{14}/r$. What is r?

$x = \frac{7080}{2^{14}}$

$$x = \cfrac{1}{2 + \cfrac{1}{3 + \cfrac{1}{5 + \cfrac{1}{2 + \cfrac{1}{\cdots}}}}}$$

the third iter has a partial sum of 35/81, the iteration after that was a denominator that is too big. Therefore we've found that since 81 is the only multiple of 81 which is also less than 100. A quick check with wolfram alpha does check out that $(35/81)(2^{14}$ is within half of 7080

(b) Suppose you know that the integer r is less than 100 and that 2979 and 14564 are both within $\frac{1}{2}$ of integral multiples of $2^{14}/r$. What is r?

$x = \frac{2979}{2^{14}}$

$$x = \cfrac{1}{5 + \cfrac{1}{2 + \cfrac{1}{1489...}}}$$

and

$z = \frac{14564}{2^{14}}$

$$x = \cfrac{1}{1 + \cfrac{1}{8 + \cfrac{1}{455...}}}$$

for both x and z we can see that the partial sum of a_2 is too big. For x we get $a_1 = \frac{2}{11}$ which checks out in wolfram alpha for $\frac{2}{11}(2^{14})$ to be within half of 2979. For z we get $a_1 = \frac{8}{9}$ which also checks out that $\frac{8}{9}(2^{14}$ is within half of 14564. From here we can say that r must be 99 since $[9, 11] = 99$ (and is infact less than 100)

Problem 6: Number of gates
(a) Fig. 3.1 shows the gates required to implement U_{FT} fro $n = 4$ Qbits. How many Hadamard operations H are required, and how many phase gates V are required? What is the total number of gates? For $n = 5$, how many H's and V's, and total gates would be required? For arbitrary n, how many would be required? (In each of the above, total gates just means H's plus V's, with each V considered a single gate.)

With $n = 4$ we need 4 H gates and 6 V gates giving us a total of 10 gates in all. With $n = 5$ we need 5 H gates and 10 V gates giving us a total of 15 gates in all. For an arbitrary n we need n H gates and $\frac{n(n-1)}{2}$ V gates so we need

$$\begin{aligned} & n + \frac{n(n-1)}{2} \\ &= \frac{2n + n^2 - n}{2} \\ &= \frac{n^2 + n}{2} \text{ gates} \end{aligned}$$

(b) After eq 3.63, it was argued that eliminating all V gates connecting Qbits more than $l = 22$ wires apart permitted factoring a 500 decimal digit number with at most a 1% loss in the probability of learning the period. How large should l be to permit factoring the largest of the RSA numbers, with 617 decimal digits (i.e., of order 100 quadrillion (googol)^6) with no more than a 1% loss in the probability? How many gates would be required (for the quantum Fourier transform part) for factoring the 617 decimal digit RSA number, eliminating all gates above the threshold value of l?

using the model example of $1/2^l < 1(500n\pi)$, we might need n to be around 4000 (the text references 3000 for a 500-digit N) for a 617 digit N. Plugging these numbers wofram spits put an l slightly larger than 22.

(c) For fun, estimate how long it would take to factor such a number using conventional techniques. The largest RSA number factored has 232 decimal digits, and was factored in late 2009, as described here (link). This took on the order of 10^{20} operations in roughly two years, hence corresponded to the equivalent of using roughly 1000, 2.2GHz single core processors over that time-frame. Assuming that the number of operations to factor a number with d decimal digits scales as $C\exp(\beta d^{1/3})$ (where C and β are some constants), and using the estimate in the above reference that a 309 decimal digit number would take about 1000 times longer, how long would it take to factor a number with 617 decimal digits using the same technology?

plugging in the system

$$Ce^{\beta 232^{1/3}} = 2 \text{ years}$$

$$Ce^{\beta 309^{1/3}} = 2000 \text{ years}$$

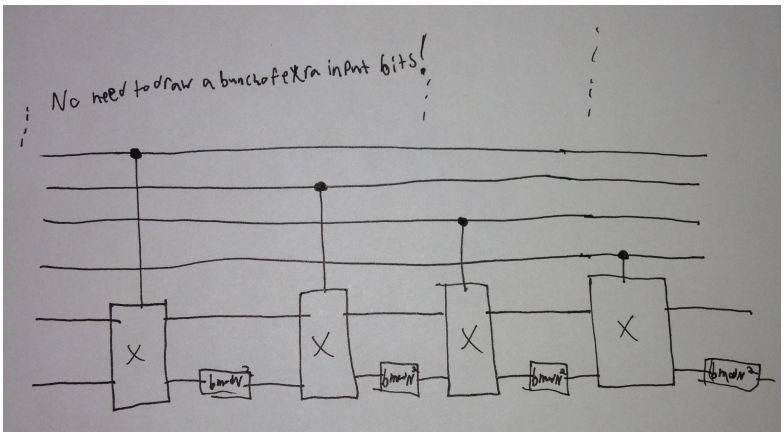
we get $C = 2.4 \times 10^{-30}$ and $\beta = 11.21$. Now calculating

$$2.4 \times 10^{-30} e^{11.21 \cdot 617^{1/3}}$$

we get that it would take 6.71×10^{11} years to finish. 6.7 hundred billion years is a pretty long time...

Problem 7: In sec 3.8 of the text (p.83-84), it's shown how to efficiently calculate the periodic function $f(x) = b^x \bmod N$, highlighting that it can be done with a single execution of the procedure that does the multiple squarings (without need of a classical look-up table).

(i) Draw a circuit diagram for how this would be done for $N = 15$, in terms of black box quantum circuits that implement controlled multiplication (step a), and mod N square (step b).



(ii) As mentioned in class, the modular exponentiation to calculate the periodic function itself might be more gate-intensice than the Fourier transform (to calculate its period). Estimate how the number of operations scales in the number of digits of the number to be factored, and roughly how many total gates (modular exponentiation plus quantum Fourier transform) might be necessary to factor the largest of the RSA numbers (as considered in problem 6b above).
Note: the details will depend on the implementation, this is just to estimate some rough number

After doing a little research into the matter it looks like it should be polynomial time. (ie the time taken is polynomial in log N, which is the size of the input). A quick search puts it at $O((\log N)^3)$. The efficiency

is mainly due to the QFT and modular exponentiation by repeated squarings.

Now for an implied complication of this problem it is true that using repeated squares for modular exponentiation requires a fair number of Qbits and many more gates than the Fourier transform. It looks like the circuit implementation for the QTF involves $q(q-1)/2$ gates.

*(Talked about a few of the problems with Andrew Casey (and a couple other kids, but mainly Andrew))