Math 406 Section 8.4: Public Key Encryption and RSA

1. **Introduction:** The primary problem with a technique like an exponentiation cipher is that given (p, e) it's easy to find (p, d) (recall $\gcd(e, p-1)$ guarantees we can find d with $ed \equiv 1 \bmod p-1$ via the Euclidean Algorithm, which is fast). Likewise given (p, d) it's easy to find (p, e).

   The primary reason this is a problem is:

   If Bob wants to receive messages and decrypt them using (p, d) then anyone who wants to send them to him must use (p, e) but then any of those people can find d and therefore can decrypt anything else that's sent to him.

   It would be great if Bob could make (p, e) publicly available so anyone could use it to send him messages and yet still have it difficult to calculate (p, d) so that messages sent to him are only decryptable by him.

   We can do this but we have to stop using a prime p and tweak the approach a bit.

2. **RSA Algorithm**

   (a) **Encryption:** Bob picks two distinct large primes p and q and calculates $n = pq$. This will be his modulus. He then chooses e with $\gcd(\phi(n), e) = 1$. Note that $\phi(n) = \phi(pq) = (p-1)(q-1)$ so he can choose an e pretty easily via the Euclidean Algorithm. Bob makes the pair (n, e) publicly available.

   Alice takes her message and breaks it up just like with the exponentiation cipher, breaking it into blocks with numerical value not possibly more than n. For each plaintext block P she calculates the ciphertext block as the least nonnegative residue:

   $$C \equiv P^e \bmod n$$

   She then sends all the ciphertext blocks to Bob.

   Example: Suppose Bob chooses $p = 59$ and $q = 73$. Then $n = (59)(73) = 4307$ and $\phi(n) = (58)(72) = 4176$. He then chooses $e = 7$ with $\gcd(e, \phi(n)) = 1$.

   Alice wishes to encrypt and send `WORD`. She divides it into blocks of length 2 and does:

   $$\begin{array}{cccc} & \texttt{WO} & \texttt{RD} & \\ & 2214 & 1703 & \\ & 2214^7 & 1703^7 & \\ \hline \equiv & 3918 & 1655 & \bmod 4307 \end{array}$$

   She sends `3918 1655`.

   (b) **Decryption:** Since Bob knows $\phi(n) = (p-1)(q-1)$ he can easily find d with $ed \equiv 1 \bmod \phi(n)$ (so $ed = 1 + k\phi(n)$ for some $k \in \mathbb{Z}$) Then for each ciphertext block C he can decrypt by calculating the least nonnegative residue $C^d \bmod n$.

   The reason for this is as follows:

   If $P \equiv 0 \bmod p$ then:

   $$C^d \equiv (P^e)^d \equiv 0 \equiv P \bmod p$$

   whereas if $P \not\equiv 0 \bmod p$ then

   $$C^d \equiv (P^e)^d \equiv P^{k\phi(n)+1} \equiv P^{k\phi(n)}P \equiv P^{k(p-1)(q-1)}P \equiv (P^{p-1})^{k(q-1)}P \equiv (1)^{k(q-1)}P \equiv P \bmod p$$

where $P^{p-1} \equiv 1 \bmod p$ by Fermat's Little Theorem since $p \nmid P$.

The same holds for q, mutatis mutandi, and so we have $C^d \equiv P \bmod p$ and $C^d \equiv P \bmod q$ and so together we have:

$$C^d \equiv P \bmod n$$

Example: Using the above knowledge Bob has calculated $d = 2983$ which satisfies $de \equiv 1 \bmod \phi(n)$.

Bob receives the message `1611 0430 2088` from Alice. He does:

	1611	0430	2088	
	1611^{2983}	0430^{2983}	2088^{2983}	
$\equiv$	0704	2401	1401	mod 4307
	`HE`	`YB`	`OB`	

So the message is `HEY BOB`.

(c) **Security:** Can anyone else find d easily? Well to find d means to solve $ed \equiv 1 \bmod \phi(n)$ which is done by the Euclidean Algorithm. The Euclidean Algorithm is easy but finding $\phi(n)$ is not.

Why not? Well (n, e) are public so how hard can it be to find $\phi(n)$? We know that $\phi(n) = (p-1)(q-1)$ so factoring n would be sufficient, and this is where the problem lies. Factoring large numbers is very difficult, and so even given n, finding p and q is no easy task.

You might ask, however, if there is a way of finding $\phi(n)$ without finding p and q. Well consider that if you did know n and $\phi(n)$ then observe that:

$$p + q = pq - (p-1)(q-1) + 1 = n - \phi(n) + 1$$

and observe that:

$$p - q = \sqrt{(p-q)^2} = \sqrt{(p+q)^2 - 4pq} = \sqrt{(n - \phi(n) + 1)^2 - 4n}$$

It follows that since:

$$p = \frac{1}{2}((p+q) + (p-q))$$

and

$$q = \frac{1}{2}((p-q) - (p-q))$$

we can find both p and q in terms of n and $\phi(n)$.

The practical upshot of this is that finding $\phi(n)$ is algebraically equivalent to finding p and q. Since factoring is hard, the problem is hard.

(d) **Digital Signatures:** Suppose Alice has public key (n_A, e_A) and private key (n_A, d_A) while Bob has public key (n_B, e_B) and private key (n_B, d_B).

If Alice just wants to send a message to Bob she uses (n_B, e_B) as discussed, but what if she wants to sign it so that Bob knows that she sent it, and that nobody else could have. What she does is first she calculates the least nonnegative residue:

$$S \equiv P^{d_A} \bmod n_A$$

to produce a *signed plaintext*. Since only she possesses d_A, only she can do this. She then encrypts:

$$C \equiv S^{e_B} \bmod n_B$$

and sends C to Bob.

When Bob receives it he first calculates S via:

$$S \equiv C^{d_B} \bmod n_B$$

which results in a signed message. He then "unsigns it" using Alice's *encryption* key:

$$P \equiv S^{e_A} \bmod n_A$$

Assuming he gets something sensible he knows for sure that Alice signed it.

Note: Alice may need to re-block the text here. This is because the result of signing a block might result in a signed block with a numerical value larger than Bob's encryption modulus.

3. **Closing Notes**

   (a) This is a very simplistic description of RSA and signatures. In real-world implementation there are various other stages involving padding, cryptographic hashes and certificates involved.

   (b) The primes are usally very large.

   (c) In trying to find large primes, how do we know we have primes when we're arguing that it's hard to factor large numbers? The answer is that generally numbers are analyzed using certainly probabilistic techniques until it seems highly probable that we have primes.