RSA Cryptography

The 1970's were an important time in the development of the science of computing. At the start of the decade Cook and Levin proved that there are some problems that are so hard that finding fast algorithms to solve them would be equivalent to giving our computers magical powers. From the point of view of cryptographers, this class of problems was an enticing opportunity. Would it be possible to create a pair of functions E and D with the desired properties mentioned above, and the added property that Eve would have to solve one of these tremendously hard problems to break the code?

Around 1977/1978, three MIT researchers named Rivest, Shamir and Adleman figured out a way to create an encryption/decryption method based on modular arithmetic. Their biggest challenge was in finding a difficult-to-solve problem that could be adapted to cryptography. In the end they hit on a very simple-sounding one: factoring.

Factoring seems easy because in every-day life we only try to factor small numbers. Numbers with 2, 3 or 4 digits can be factored with just pencil and paper (and a handy list of primes). For numbers with a few more digits, we can write a trivially simple program that will just try all the possible factors – our computers are pretty fast and we get the results quickly.

But if we are dealing with integers that have *hundreds* of digits, this "try all the possible factors" method of factoring is completely infeasible – because dividing one really large number by another is a very slow process (just remember how time-consuming long division is, compared to multiplication).

RSA realized that if we choose two *really big* primes p and q and multiply them together to get $n = p * q$, we can tell people the value of n but nobody will be able to figure out p and q. (Because **factoring is hard.)**

The idea is for Bob to create an E function that can be made public (so both Alice and Eve can see it), and the corresponding D function that Bob keeps secret, *which cannot be figured out by somebody who knows* E. These two functions will make it possible for Alice to send messages securely to Bob.

To turn this "factoring is hard" idea into a pair of functions E and D, we need to introduce a bit of notation:

Let $n = p * q$. Then we define $\phi(n) = (p-1) * (q-1)$

The ϕ symbol is the Greek letter ***phi.*** In the Greek language it is pronounced "fee" but most mathematicians I have met pronounce it "fie" (rhymes with "try")

Example: Let $p = 7$ and $q = 13$. Then $n = 91$, and $\phi(n) = 6 * 12 = 72$

Leonhard Euler was one of the most prolific and important mathematicians in history. Among many other important results, he proved this theorem about $\phi(n)$:

Theorem: Let n and $\phi(n)$ be defined as above, and let a be a number that is relatively prime to n. Then

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Equivalently,

$$(a^{\phi(n)})\%n = 1$$

This result is crucially important to our discussion, so let's explore it a bit. Remember that $\phi(n)$ is just a number that we compute from n. In the example above we had $p = 7, q = 13, n = 91, \text{ and } \phi(n) = 72$

So this theorem tells us that if a is any number that is relatively prime to 91, then

$$(a^{72})\%91 = 1$$

Now the only factors of 91 are 7 and 13, so any number that does not have either 7 or 13 as a factor will be relatively prime with 91.

For instance, we automatically know that $(22^{72})\%91 = 1$, and $(61^{72})\%91 = 1$, and $(85^{72})\%91 = 1$, because none of those numbers are divisible by 7 or 13. (I just picked 22, 61 and 85 at random.) Most numbers in the set $\{1, \ldots 90\}$ are relatively prime with 91. (The exact answer is that 72 numbers in this range are relatively prime with 91 … and as we have just seen, $\phi(91) = 72$. This is not a coincidence, but proving it is not important to us right now!)

The function $\phi(n)$ is often called **Euler's totient function.**

Now we're going to use $\mathbb{Z}_{\phi(n)}$ to pick two numbers that we will use to define E and D.

We want to find numbers e and d such that e and d are inverses in $\mathbb{Z}_{\phi(n)}$. In other words, $e \otimes d = 1$ in $\mathbb{Z}_{\phi(n)}$. In our standard notation, $d = e^{-1}$ in $\mathbb{Z}_{\phi(n)}$

Finding e and d can be a non-trivial task when $\phi(n)$ is very large, but we only have to do it once. There are algorithms for computing inverses but they are outside the scope of this course.

It turns out that we can use $x^e \% n$ as our encryption, and $x^d \% n$ as our decryption – good thing we studied exponentiation in modular arithmetic!

So now we have $e \otimes d = 1$ in $\mathbb{Z}_{\phi(n)}$. We know this means $e * d = k * \phi(n) + 1$

The next few steps are the essential core of the justification of RSA Cryptography.

Suppose $a < n$ and $gcd(a, n) = 1$ (ie. a relatively prime with n). Then

$$\begin{aligned}
(a^{e*d})\%n &= (a^{k*\phi(n)+1})\%n \\
&= (a^{k*\phi(n)} * a)\%n \\
&= ((a^{\phi(n)})^k * a)\%n \\
&= ((a^{\phi(n)})\%n)^k * a)\%n \\
&= (1^k * a)\%n \qquad \text{by Euler's useful theorem!} \\
&= a\%n \\
&= a
\end{aligned}$$

This is fantastic!

$$(a^{e*d})\%n = a$$

So we start with a, raise it to a power, take that $\%n$, and we end up with a again. Can we split the mathematical process into an E and a D function? Yes! Because $(a^{e*d})\%n$ can be computed as a two-step process:

- first compute $y = (a^e)\%n$
- second compute $z = (y^d)\%n$

By what we have just shown, $z = a$!!!

Let a be the message that Alice wants to send, and assume that a is an integer such that $a < n$ and a is relatively prime with n (we'll address the reasonability of those assumptions a bit later in these notes).

Let $E(a) = (a^e)\%n$ and let $D(y) = (y^d)\%n$

I claim that these two functions have precisely the properties that we need:

- D exactly "undoes" what E does, so it decrypts the encoded message
- even if Eve knows E, she cannot figure out D (so she can't decrypt messages)

We have already seen that the first claim is true: $D(E(a)) = a$

For the second claim, let's think about what Alice needs to know in order to send her message to Bob. She needs to know e and n, but she doesn't need to know $\phi(n)$ or d. Bob can post e and n in a public location.

This means Eve knows the same things that Alice knows: e and n. Not only that - Eve can also see $E(a)$ when Alice sends it to Bob. But to decrypt the message she needs to know d. She knows that $d = e^{-1}$ in $\mathbb{Z}_{\phi(n)}$ but she doesn't know $\phi(n)$

Eve doesn't know $\phi(n)$ because Bob keeps it secret. Now if Eve knew p and q, she could easily compute $\phi(n) = (p-1) * (q-1)$ …. but Bob keeps p and q secret too. Eve could find p and q if she could factor n … **but factoring is hard!!!!!** So Eve is defeated. She (like Alice) knows how to encrypt and send messages to Bob, but she doesn't know – and can't figure out – how to decrypt messages that are addressed to Bob.

So RSA cryptography is secure.

Note that if anybody ever discovers a fast algorithm for factoring very large numbers, RSA cryptography will no longer be secure – and the world of e-commerce will come to a sudden halt.

A few final thoughts on this before we do an example. So far all we have done is work out a secure way to send messages to Bob. If Bob wants to send secure messages to Alice, she has to do the same things he did:

- pick two large primes, p_A and q_A (I'm giving them A subscripts to show they belong to Alice)
- compute $n_A = p_A * q_A$
- compute $\phi(n_A) = (p_A - 1) * (q_A - 1)$
- find e_A and d_A such that $(e_A * d_A)\%n_A = 1$
- publish e_A and n_A

Now Bob (and anyone else) can send messages securely to Alice. If Bob wants to send message T to Alice, he computes and sends $X = (T^{e_A})\%n_A$. When Alice receives X, she computes $(X^{d_A})\%n_A$ which turns X back into T

In general, if a group of people all want to send encrypted messages to each other using RSA they each need to design their own E and D functions, and they each need to publish their individual e and n values. Anyone wishing to send a secure message to person i just looks up e_i and n_i and uses those to encrypt the message. Then person i uses their secret d_i to decrypt it.

Now what about the assumption that the message a is an integer? We can convert any text string to an integer by converting each letter to a two-digit number (for example A = 01, B=02, etc.) and then just stringing the numbers together. So "ZEBRA" might become 2605021801. We also assumed $a < n$. If we are dealing with a long message that converts to an integer $\geq n$, we can simply split the message into smaller pieces and encrypt them separately. We also assumed that a is relatively prime with n. This is a very safe assumption because when p and q are large primes, the number of elements of $\mathbb{Z}_n$ that are **not** relatively prime with n is very tiny compared to n. However it turns out that even if we are unlucky and $gcd(a, n) \neq 1$, the encryption/decryption will still work (we'll skip the proof of this – it's harder).

Finally, let's do a full example.

Suppose Bob chooses $p = 11$ and $q = 13$ (remember, in real use these primes would be hundreds of digits long)

$n = p * q = 143$

$\phi(n) = (p - 1) * (q - 1) = 120$

Now Bob chooses e in $\mathbb{Z}_{120}$ such that e has an inverse. He just has to ensure that e is relatively prime with 120 … easy enough to do since $120 = 2 * 2 * 2 * 3 * 5$. We'll say Bob chooses $e = 37$.

Bob has to compute 37^{-1} in $\mathbb{Z}_{120}$ … which turns out to be 13, so $d = 13$

Bob publishes the information $e = 37, n = 143$

Suppose Alice wants to send the message 75 to Bob

She computes $(75^{37})\%143$ and gets the result 114

(I used a spreadsheet – but since we studied how to do exponents in modular arithmetic you know two different ways to do this!)

Alice sends 114 to Bob.

Eve is frustrated – she knows she needs d to decrypt this, but she can't figure it out. She can't factor 143 (ok, factoring 143 is actually pretty easy … but if this were a huge integer it would not be easy at all) so she cannot figure out p and q, which she needs in order to compute $\phi(n)$ … which she needs in order to figure out d.

Bob computes $(114^{13})\%143$ …. and gets 75 !!! Ta-dah!!!!!

Bob never has to change his e and n values unless he is worried that Eve might somehow have discovered $\phi(n)$

Note that Alice and Bob don't have to use the same values of n, or anything else. All they have to agree on is they will both use RSA encryption.

So, are there any downsides to RSA? Unfortunately, yes. Even though we have studied ways to compute $a^e \% n$ efficiently, it is still true that we are dealing with really big numbers. So both the "repeated squares" method and the "find cycles" method may take quite a while to compute the encrypted form of the message … and decryption will also be slow. For a lot of communication (email, for example) that may not be important. But if we are trying to stream live video and/or audio the delay may not be acceptable.

Fortunately there is a compromise. There are lots of faster encryption methods that are quite strong as long as Eve doesn't know either E or D. (Remember, the strength of RSA is that Eve can't figure out D even though E is public knowledge.) In the popular PGP (Pretty Good Privacy) encryption method , Alice and Bob start their conversation by using RSA (or another similar public-key method) to agree on an E, D pair of functions. They then use E and D to encrypt the rest of the conversation. Since Eve can't read the first part of the conversation she doesn't know what E and D are, so the rest of the conversation is quite secure. PGP was first developed in 1991 – its history makes for fascinating reading. Over the last three decades it has developed a lot and is now extremely powerful. Many email clients include a built-in version, and the Free Software Foundation has developed an open-source version of PGP – which is confusingly called GPG.