Explain it like I'm 2: ElGamal

Sam Nolan

May 2, 2018

ElGamal is an encryption algorithm that solves a few problems with RSA. The algorithm is secure because of the difficulty in calculating a discrete logarithm.

Here, we will try and use a rather peculiar metaphorical use of ElGamal in order to explain the mechanics of how it works. I would recommend that you have read *Explain it like I'm 2: RSA*, as reading it will give you a deeper understanding of how ElGamal works. However, it is not necessary, and you can understand ElGamal without knowing RSA.

There is pretty much no new maths for this paper.

The kings vote

This metaphor goes through the ElGamal process and the problem that it solves.

A king has recently been advised by his subordinates that it should be time that the country becomes a democracy. As of such, the king would like to hold a vote.

Everyone in his country will be able to vote. But he wishes that no one but the government knows of who voted for who, so he wants a system where the voter can encrypt their vote in some way that only the government can unlock.

There is also one other problem, as there will be multiple people voting for the same person, if two people come up with the same ciphertext, then they can find out who the other person voted for.

So we have 2 requirements of the cryptosystem:

- The message must be encrypted in such a way that only the king can unlock it
- Different people voting for the same person should get different ciphertexts

Now, as we know, RSA successfully fulfils the first requirement, but not the second, if 2 people are to encrypt the same message in RSA with the public key, they will get the same message.

So what's a king to do? Well luckily, one of his advisers is an expert in cryptography systems named ElGamal, and he comes up with a rather peculiar scheme.

The public key

ElGamal advises that they will distribute a box that has three items on it. The three items are the message wheel, a unset generator, and a set generator.

The message wheel

The message wheel is like a smaller version of a circular prize wheel, it has all the faces and names of the people that you can possible vote for printed around the edges, with a pointer at the top indicating a particular person.

The message wheel can discretely point to one person and the pointer wont move from person to person when it's in travel.

If we were to have an unencrypted scheme, a voter would turn the wheel until the pointer is pointing towards the person that they want to vote for, and give the wheel to the voting agency. However, this wheel will be used a bit more intelligently for our scheme.

A generator

A generator is a peculiar device that is similar to the message wheel, but instead of faces and names, it simply has unique numbers around the edges that are in a random order.

The device has been locked down so you can only see one number, at the top, and it has a clicker on the side that rotates the generator anti-clockwise, giving you a new number.

As part of the key, you are given 2 generators, one that has been set, and one that has not been set. An unset generator is one that is simply displaying the number 1, a set generator has been clicked the amount of times that is equal to the king's private key.

The king's private key is simply some random number. The only thing that's special about it is that only the king knows about it.

Mapping this to key generation

Key generation in ElGamal uses the exact same process.

First, we have g and p, for which p is a prime number. g and p make up the description of the generator above, where p is the amount of different numbers in the generator and g determines what order that the numbers will be in.

To find the n^{th} number on our metaphorical generator, we use the following formula:

$$g^n \mod p$$

Do you remember our 7 times tables back from the RSA paper? This is a very similar idea. g is chosen so that every possible number between 0 and $p-1$ can result from different values of n. As with the metaphorical generator above, it creates a random permutation of the numbers from 0 $p-1$.

An unset metaphorical generator is the same as a description of the generator derived from g and p, a set one has had the king apply his key to it. So if the king chooses a random number x, then the king clicking the generator x times is analogous to:

$$y \equiv g^x \mod p$$

We'll call this set generator y.

The voter's encryption

The box with the three peculiar items also comes with a small instruction set. The instruction set is as follows:

1. Turn the message wheel until it is pointing towards the person you want to vote for
2. Choose a random number r and remember it
3. Pick up the unset generator in one hand the set generator in the other
4. Click both generators r times
5. Put the (previously) unset generator aside and look at the number that is on the double-set generator (We will call the shared secret (or s) from now on)
6. Move the message wheel anticlockwise the amount of times as displayed by the shared secret
7. Burn the shared secret, or find some other way to hide or destroy it
8. Put the message wheel back into the box along with the (previously) unset generator that you put aside
9. Give this box to your nearest voting agency

ElGamal encryption

Let's go through all these steps in ElGamal.

First, turn the message wheel to the message that you want to say. This is analogous to mapping your message onto a particular number. For instance, using ASCII, you can map any string of characters into a number that you can send.

Then, choose a random number and remember it. We will call this number r.

Then, pick up both generators and click them according to your random number. Remember, we have called the generator set by the king y which is equal to g^x. An unset generator is simply represented by g.

To set both generators with your random number, you can bring both numbers to the power of r.

For the unset generator g, "setting" it by your random number r makes simply g^r. We will call this C_1. C_1 is simply used as a description of the random number that you have chosen. We're calling it C_1 because it will be sent as part of the ciphertext later.

As for the set generator y, this becomes y^r or g^{rx}. We will call this number s, as it is the shared secret.

You can then move the message wheel x spaces, this is represented as

$$C_2 \equiv s \times m \mod p$$

This text is ciphertext 2. It actually contains the message within it.

Now, if someone has access to s, they can invert the encryption process. So s is destroyed or hidden, never transmitted.

The ciphertexts are then both sent to the other party.

The king's decryption

Now the king has both the message wheel that has been spun as per the shared secret, as well as the generator that has been clicked r times.

Now, if the king clicks the received generator x times, it will be the same as the shared secret that was made by the voter.

The voter got the shared secret by getting a generator that was clicked by the private key and clicking it a random number of times.

The king got the shared secret by getting a generator that was clicked a random number of times and then clicking it by his private key.

Both methods will get the same number.

The king then moves the message wheel clockwise (the other way) by the number on the shared secret, and he will get the original person that the voter voted for!

ElGamal Decryption

The receiver has now received g^r or C_1 as ciphertext as well as $g^{rx}m \mod p$ which is $sm \mod p$ or C_2.

To recreate the shared secret, the receiver can use his private key x and take $C_1^x \mod p$, which is the same as $g^{rx} \mod p$ or s.

To spin the message wheel the other way around, the receiver simply uses the multiplicative modular **inverse**. That is

$$m \equiv C_2 \times s^{-1} \mod p$$

and there we have it! we have the message back! Because of the random number generated by the voter, we will get different ciphertexts from the same message. And only the king can decrypt the ciphertext!

Limitations on a metaphor

The metaphor is not bad, but like all metaphor's, it's not perfect.

One good thing about the metaphor is that it demonstrates the circular nature of modular arithmetic. Modular arithmetic repeats itself over in a similar fashion to the message wheel and the generator.

Also, the metaphor shows that the best way that someone can find the private key, is to get an unset generator and keep clicking it until you find the number that was on the king's generator.

Although in the metaphor, this would be entirely feasibly, in real life, the size of the generator p is so extraordinarily large that this in no longer feasible.

Another important distinction to be made is the difference between operations that are easily reversible and operations that are difficult to reverse.

In ElGamal, an operation that is difficult to reverse is the power operation. As if you know g and g^x it is difficult to find x. This is at the heart of the security of ElGamal encryption, as both g and g^x are part of the public key.

On the other hand, a difficult that is easy to reverse is multiplication, if you know s and sm it is easy to find m. This is what the king does in his final decryption step. What makes it easy is because there is a known algorithm for solving this sort of problem knows as the *Extended Euclidean Algorithm*. This is what online calculators use to calculate the modular multiplicative inverse.

In the metaphor, this was represented by the message wheel being an open and free wheel, where it was easy to find what you are looking for and move backwards and forwards, and a locked down "generator" which can only be clicked forwards.

However, this is not quite how ElGamal works as it is not that the generator is used that makes it hard to reverse, it's the **operation** that makes it so. A slightly better metaphor would be the following:

1. You are given three completely open disks. One of which has faces on it, and numbers under those faces. The other two just have numbers. One of the disks with just numbers is set to 0, and the other is set to another number (y)
2. Choose a random number r
3. Get the disk that is set to 0 and set it to 1
4. Repeat the following r times
   (a) Multiply the number that is displayed on the disk by g (Some number also in the box)
   (b) Whatever results, move the disk to that number, if the number is larger than the disk, mod the number by the size of the disk and move it to the result
5. Pick up the disk starting at a number that is not 0 (y) and repeat. The result is called s

6. Remember the number under the face you want to vote for as m
7. Set the face wheel to the face set at number 0
8. Repeat the following s times
   (a) Move the wheel m times clockwise
9. Destroy the wheel with s on it, and send the other two wheels

As you can imagine, if the voter was to go through this process, they would likely not even bother voting! This is similar to the process run by computers in ElGamal.

You can see that moving the wheel by multiplication (as of the last loop) is much more orderly than moving the wheel by powers (as of the first loop). You can imagine going from one number to the next with seemingly no pattern, where as the iterations made with multiplication is still a bit chaotic but a bit more orderly.

Now imagine getting the result from the first loop, knowing g, would you have any hope in finding how many times it has been rotated? I certainly wouldn't.

What I am trying to exemplify here with the more accurate metaphor is that it's not the wheels that are different, but the operation that is different.

To match these two versions of the metaphor, you can think of the metaphorical generator doing the contents of the first loop automatically, so that every iteration is simply done by clicking on a button. It would make the process less of a nightmare for the voter.

Notes

ElGamal Signature Scheme

Unlike RSA. The only thing that relates the ElGamal signature scheme and ElGamal encryption is that is was made by the same person. The signature scheme is not the reverse of the encryption scheme.

OK, that's not entirely true, there are similar elements between the two. But one is not the reverse of the other as RSA is. Explaining the ElGamal signature scheme would be a topic for another paper.

Exponential ElGamal

Exponential ElGamal is the same as ElGamal except that g^m is encrypted rather than m. This means that on the decryption side, the person will receive g^m and because this is difficult to undo, will have to guess values of m until they find one where g^m is the same as the value received. This is all OK if m is small, but gets difficult the larger m gets.

So what's the point of making it harder? Well, Exponential ElGamal is additively homomorphic, meaning that it's possible to add the encrypted numbers without knowing what the numbers are. This is part of privacy preserving computations. For which is outside the scope of this document.

I hope you found this useful! If you have any feedback or you think that another topic needs some justice. You can either contact me at s3723315@student.rmit.edu.au or submit an issue at github.com/Hazelfire/EILI2.