RSA Project Report

By: Taher Mohamed
Peter Michael

Key generation

In this project we generated RSA key pair by:

1. Generating random integers
2. Testing if the generated integer is prime
3. If it is not prime go to step 1

We did this for p, q, e then n = pq , Φ(n) = (p-1)(q-1), de = 1 (mod Φ(n)), d is calculated using "extended euclid algorithm".

To generate n-bits key modulus (n) we generate random integers p, and q of size n/2 bits (this doesn't lead to exactly n-bits n) but it is around n-bits

To check if a number is prime (primality test), we used Fermat's primality test based on Fermat's little theorem:

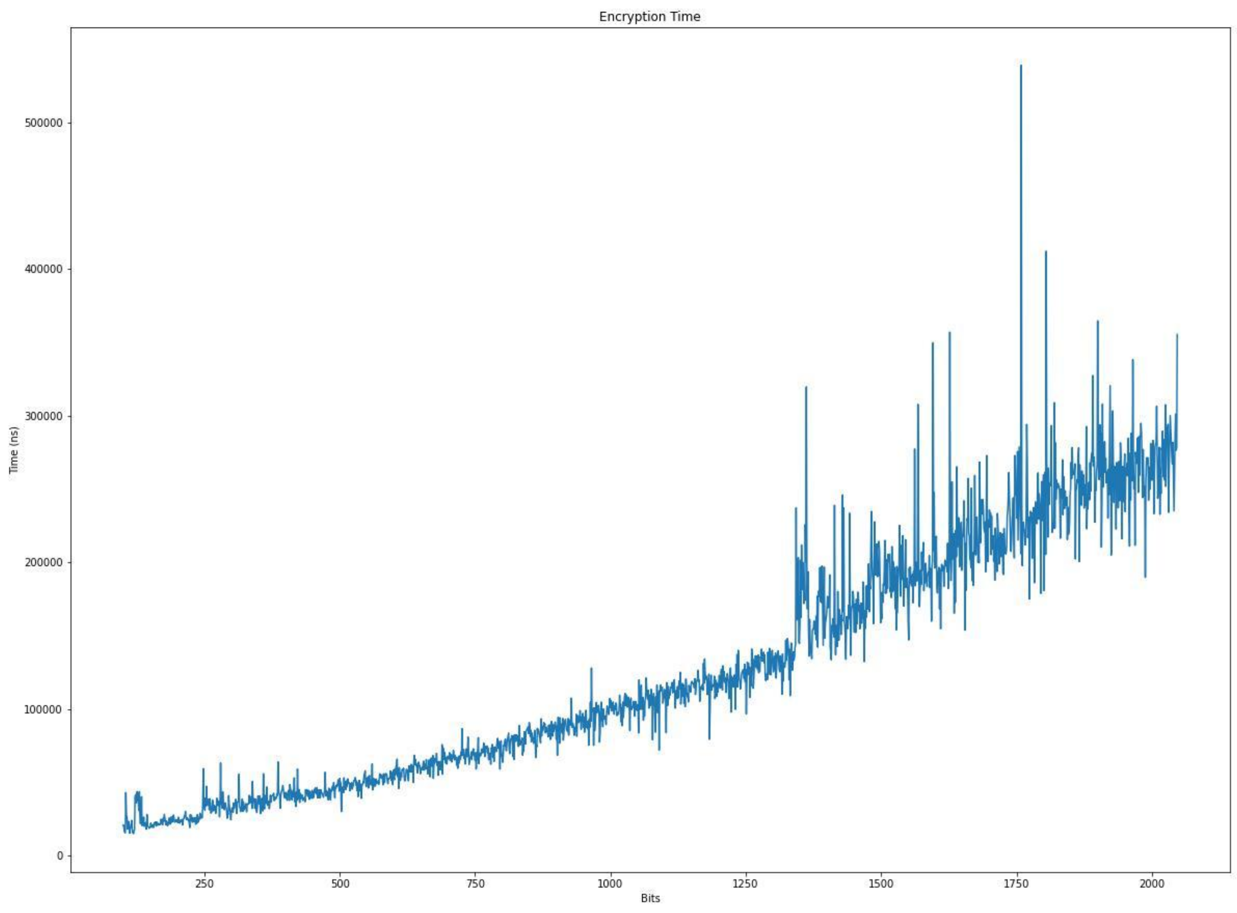
$$a^p \equiv a \pmod{p}.$$

If the number satisfies this relation for any "a" then its a prime

Encryption and Decryption

Encryption is achieved by:

1. Dividing the text to the maximum block size possible that doesn’t result in a message with numerical value bigger than n
2. Each block is converted from a string to a number by multiplying characters by 256 and adding to the result iteratively (256 is the maximum value of one byte)
3. The message is powered to the public key e using power modulus operation
4. The resulting number is padded with one or two zeros depending on n and its length to form a fixed size block to make decryption easier



Comment: neglecting random errors, we can see that the curve is growing quickly which means that RSA is a slow encryption algorithm which should be used for small encryption blocks.

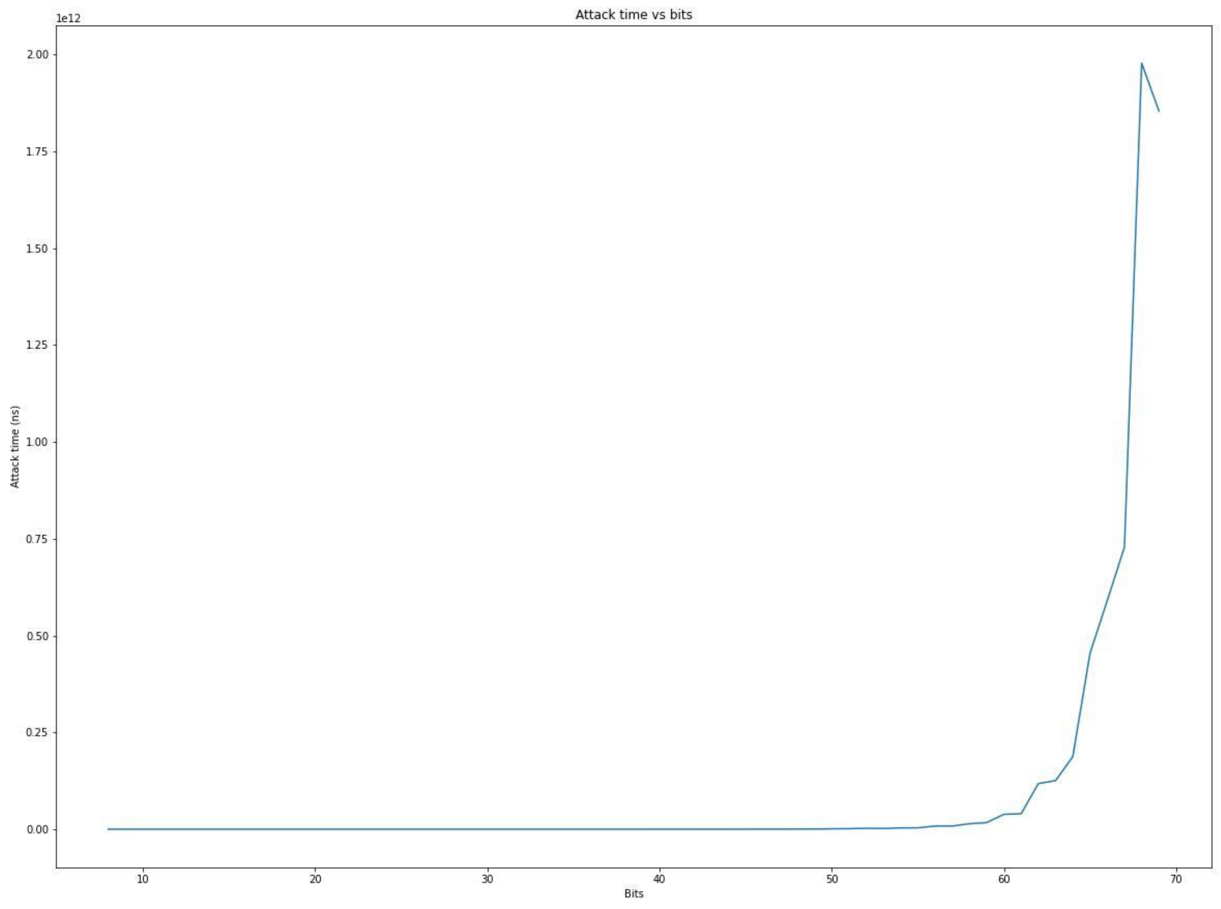
Decryption is similar to encryption:

1. Dividing the received message to fixed size blocks
2. Each block is powered to the private key using regular power modulus operation
3. The resulting number is converted to string by using division operation by 256 where the remainder is the ascii value of the character and the quotient is rest of the string

Brute Force Attack

Brute force attack is done by iterating over all numbers that are less than n and trying to divide n by these numbers. When reaching a number that divides n , then assign this to p and the division result is q.

When we reach p and q we can simply calculate d from the following equations.
$\Phi(n) = (p-1)(q-1)$
$de = 1 \pmod{\Phi(n)}$
d is he mod inverse of e in mod(Φ(n))
which is the mod inverse of e



Comments : the process of breaking the size of the key is exponential which makes it computationally infeasible to break the key for very large key sizes.

Chosen Ciphertext Attack

In this attack we are interested in getting the plain text related to a certain cipher text.

Let the cipher related to the plaintext M be called C1.

To perform this attack we must have C1 and have the ability to make the person (X) who received C1 decrypt another ciphertext C2 and send the result (Y) back to us.

After getting Y we can obtain the original plaintext P as follows.

$C1 = M^e \pmod{n}$
$C2 = C1 * r^e \pmod{n}$
$Y = C2^d \pmod{n}$
$Y = (C1 * r^e)^d \pmod{n}$
$Y = (M^e * r^e)^d \pmod{n}$
$Y = M * r \pmod{n}$ since e, d are inverses
$M = Y * r^{-1} \pmod{n}$

r is a random number such that $\gcd(n, r) = 1$, this is a requirement to be able to obtain r^{-1}.

(e, n) is the public key of X
(e, d) is the private key of X

Conclusion

RSA is a strong encryption algorithm in terms of computational security, but it lacks the required efficiency to be widely used, thus it is better to encrypt small junks of data and keys with RSA and use these keys to encrypt bigger data.