SecureIQlab®

# Understanding Prime Numbers in the Context of PKI

Author: Sridatta Raghavendra Chintapalli

www.secureiqlab.com

# Contents

**Abstract**

Secure communication and the exchange of sensitive information are essential in the current digital era. The basis for securing privacy and authenticity in online transactions, data exchange, and communication for the most part relies upon public key infrastructure (PKI). The fascinating world of prime numbers, which looks so basic but with some amazing properties and is essential to contemporary encryption, is at the foundation of PKI. In this article, we will explore the importance of prime numbers, the idea of prime factorization, and how large primes strengthen the PKI infrastructure. We will also demystify our operations to comprehend how two crucial cryptographic systems, RSA and Diffie-Hellman, use prime numbers for communication.

# 1 Primes are the foundation of security

Mathematicians and alike from the days of antiquities have always been intrigued with prime numbers. Due to their inherent unpredictable nature and mathematical characteristics, these distinctive numbers serve as the primary building blocks of contemporary encryption. Due to the fact that they seem so basic, prime numbers play a crucial role in the development of sophisticated encryption algorithms.

## 1.1 What are Prime Numbers

In a nutshell, integers higher than 1 have no divisors besides 1 and themselves. We can also state that prime numbers are products of exactly two numbers, 1 and the number itself.

## 1.2 The importance of Prime Factorization

Many encryption techniques are based on the mathematical idea of prime factorization, which is the process of dividing an integer into its prime components. A big composite number can be created by multiplying two prime numbers together, but it is far more difficult to reverse this process and isolate the original prime factors from the composite number. To ensure the security of data transmission, encryption techniques take advantage of this capability.

## 1.3 The Strength of Large Primes in PKI

Large prime numbers are used in PKI to generate robust encryption keys. The difficulty of breaking down huge composite numbers into their prime components prevents malevolent attempts to decrypt data from succeeding. As the computing work required for prime factorization increases exponentially with the size of the primes, the encryption becomes more secure the

greater the prime numbers employed.

Example: Let's take a prime number 2996863034895 and find out which combination of prime numbers created this number.

Starting with prime numbers:

We can check pairs of prime numbers to see if their product equals 2996863034895. Starting with the smallest prime, 2. However, 2 is too small to be part of the prime factorization of 2996863034895.

Moving on to the next prime, 3. Again, 3 is too small to be a factor. Continue with larger primes: 5, 7, 11, 13, and so on. After checking various combinations, we find that the prime factorization of 2996863034895 is: 2996863034895 = 5 * 599372606979.

A conventional computer could compute this number in a matter of seconds or minutes. However, the strength of PKI lies in selecting a really large number, like the one that is indicated below.

160630393859959858403673358280589999181824270743702686766853447808
113686595032379775270855285205603517787773581452662258586298854198
088226545769625982424577024442756940367615139713401263698454836286
268191545734053746348197852653690734067448833293732555971297334238
820768817693536232555504815218981217702993799922836946058364394332
786285138585788360869153430548036327141873573881929843779500484349
960741533266926289332656487105743724325052526868801

It can take a long time to factor the above number, which has hundreds of digits, using traditional techniques like the Number Field Sieve (NFS). It could take hours, days, or even longer to factor in large numbers, even with modern hardware and effective implementations.

## 1.4 Utilizing the power of large prime numbers (RSA)

One of the most popular encryption methods in PKI is RSA (Rivest-Shamir-Adleman). It largely depends on the properties of prime numbers in math-

ematics. Two unique prime integers are chosen to create an RSA key pair. For encryption and decryption, respectively, one prime is needed. The modulus, the product of these two prime numbers, is difficult to factor, and this complexity is what gives RSA its security. It is still extremely difficult to break RSA encryption by factoring huge primes, even with modern processing capacity.

Example: Let's say Alice wishes to send Bob a secure communication. Here is an overview of the RSA procedure:

1. Bob generates a key pair: a public key (containing a modulus and an encryption exponent) and a private key (containing the same modulus and a decryption exponent).
2. Bob shares his public key with Alice, who uses it to encrypt her message.
3. Alice sends the encrypted message to Bob.
4. Bob uses his private key to decrypt the message and read its contents.

# 2 How secure is RSA

The difficulty of factoring huge numbers is directly related to the security of RSA encryption. While multiplying two large prime numbers to produce the public and private keys is relatively simple, factoring the product of those two prime numbers (the modulus) back into its prime components is thought to be computationally infeasible for sufficiently large prime numbers. This is where RSA's strength lies. The foundation of RSA's security is this computational challenge.

## 2.1 Two significant factors impact RSA's security

1. Key Size: The encryption gets safer the greater the prime numbers used to create the RSA keys. Larger key sizes are advised to keep security

as computational power increases. As of my most recent update, RSA key sizes of 2048 bits were frequently advised, and several apps had already switched to key sizes of 3072 or 4096 bits.

2. Algorithmic Development: The creation of more effective factoring algorithms has an impact on the security of RSA. Although Shor's algorithm can be executed on large-scale, fault-tolerant quantum computers, RSA is still thought to be secure against conventional computers. However, it is unclear when such quantum computers will be developed that will be able to crack RSA.As the race to gain quantum supremacy is ongoing, it's just a matter of time before such computers will be able to crack RSA for a reasonable key length for a reasonable computing price and reasonable value of the information being deciphered.

# 3 Diffie–Hellman key exchange

This technique is used for the exchange of keys between two parties securely over an insecure medium. This technique allows the sharing of the key for algorithms such as DSS which can be used for digital signature but not for key exchange.

## 3.1 Example 1

Let's say Alice wishes to send Bob a secure communication. Here is an overview of the Diffie–Hellman key exchange procedure:

Alice and Bob share a prime "p" and "a" such that $a < p$ and a is a primitive root of p. A primitive root is a number when raised to different powers such as $(a^2, a^3, a^4, a^n)$, generating all remainder when divided by a prime number.

1. Alice generates a private key $X_a$ such that $X_a < p$.
2. Bob generates a private key $X_b$ such that $X_b < p$.
3. Alice calculates a public key $Y_a = a^{X_a} mod p$.

4. Bob calculates a public key $Y_b = a^{X_b} mod p$.
5. Alice sends $Y_a$ to Bob over an insecure medium.
6. Bob sends $Y_b$ to Alice over an insecure medium.
7. Alice calculates shared secret key $K_a$=$Y_b^{X_a} mod p$.
8. Bob calculates shared secret key $K_b$=$Y_a^{X_b} mod p$.

As one can see, Alice and Bob calculated the same shared secret key using this procedure. This allows Alice and Bob to use such shared keys for subsequent symmetric encryption of messages.

## 3.2 Example 2

Here is an example with some numbers:

Let's say p=23, g(p)=5 [1]

1. Let's say that Alice's private key is $X_a = 6$
2. Let's say that Bob's private key is $X_b = 15$
3. The public key of Alice is $Y_a = 5^6 mod 23 = 8$
4. The public key of Bob is: $Y_b = 5^{15} mod 23 = 19$
5. $Y_a$ is sent to Bob and $Y_b$ to Alice.
6. Shared Key K as calculated by Alice is:$K_a$=$Y_b^{X_a} mod p$ i.e $19^6 mod 23 = 2$
7. Shared Key K as calculated by Bob:$K_b$=$Y_a^{X_b} mod p$ i.e $8^{15} mod 23 = 2$

As one can see the shared key $K_a$=$K_b$ and can be used to sign symmetric encryption of messages. Bear in mind that Diffie–Hellman key exchange is susceptible to Man-in-the-Middle Attack.

[1] https://www.geeksforgeeks.org/primitive-root-of-a-prime-number-n-modulo-n/)

## 3.3 Man-in-the-Middle Attack in Diffie–Hellman

A very simple example of a procedure for a Man-in-the-Middle Attack in Diffie–Hellman key exchange[2] is as follows. Let's say Alice wishes to send Bob a secure communication but Voldemort is in the middle to intercept the key exchange. Here is an overview of the Diffie–Hellman key exchange procedure:

$p = 23$ and $g(p) = 5$

Alice generates a private key. $X_a = 6$

Computes the Public key

$$Y_a = g(p)^{X_a} mod p = 5^6 mod 23 = 8 \tag{1}$$

and sends it over to Bob.

Alice will compute the secret key as

$$K2 = Y_d 2^{X_a} mod p = 20^6 mod 23 = 16 \tag{2}$$

Voldemort generates two private keys. $X_d 1 = 3; X_d 2 = 5$

He also intercepts Alice's and Bob's public keys $Y_a$ and $Y_b$ respectively.

Voldemort computes two public keys

$$Y_d 1 = g(p)^{X_d 1} mod p = 5^3 mod 23 = 10 \tag{3}$$

and sends it to Bob instead of Alice's public key.

$$Y_d 2 = g(p)^{X_d 2} mod p = 5^5 mod 23 = 20 \tag{4}$$

and sends it to Alice instead of Bob's public key.

Voldemort can also compute secret Keys as

$$K2 = Y_a^{X_d 2} mod p = 8^5 mod 23 = 16 \tag{5}$$

[2]Some assumptions are made such as p and thus g(p) is known to the Man-in-the-Middle party.

$$K1 = Y_b^{X_d 1} mod p = 19^3 mod 23 = 5 \tag{6}$$

Bob generates a private key. $X_b = 15$

Computes the public key

$$Y_b = g(p)^{X_b} mod p = 5^{15} mod 23 = 19 \tag{7}$$

and sends it over to Alice.

Bob will compute the secret key as

$$K1 = Y_d 1^{X_b} mod p = 10^{15} mod 23 = 5 \tag{8}$$

As one can see, Voldemort has Bob and Alices' secret key and since he is in the middle, he can send messages impersonating both Bob and Alice at his convenience. However, in the real world, the key exchange is more complex, including the sharing, big prime, and primitive root of prime.

# 4 Quantum Computing threat and the future

Quantum computers equipped with Shor's algorithm could pose a catastrophic threat to the PKI infrastructure, which is based on the security of prime numbers. Large prime numbers' current resilience to attacks may shatter in the face of quantum machines' immense processing power. The foundation of safe online interactions could be destroyed by the rapid decryption of encrypted communications and data that previously remained immune to conventional attacks. Researchers and specialists in cryptography are working against the clock to develop and put into practice quantum-resistant encryption techniques as the threat posed by these technologies grows. Data security will be maintained even in a post-quantum world thanks to these new algorithms, which are built to withstand the powerful computing power of quantum computers.

Some of the cryptography algorithms that are being investigated are:

1. Lattice-based cryptography

2. Code-based cryptography
3. Multivariate Polynomial Cryptography
4. Hash-based Cryptography

One possible solution to these problems is the QKD (Quantum Key Distribution), which utilizes the phenomenon of quantum entanglement and the no-cloning theorem to establish an unbreakable key exchange between parties, ensuring that any eavesdropping attempt is detectable.

Having said that, NIST predicts that by 2029, quantum computers will be able to crack current public key infrastructure, such as the 128-bit AES encryption now used to safeguard private data exchanged over the Internet. The uncertainty around quantum computing will, however, only be eliminated if a hardware platform with at least 10,000 qubits is made scalable and affordable.

# 5 References

[1] Garth Crosby-Cryptography Lectures.