1 Introduction

The concept of Public Key Cryptography originated around 1970 when a number of teams and research studies independently discovered similar concepts. The most notable of these researchers - although possibly not the first - were Whitfield Diffie and Martin Hellman, whose publication “New Directions in Cryptography”, provided the groundwork for public key encryption systems that we continue to use today. The concept of secure communication, even in the presence of eavesdroppers, is of timeless importance and in our modern world we utilize what Diffie and Hellman defined as a Public Key Cryptosystem (PKC). Fundamentally, a PKC employs two main components: a one-way-function - an invertible function whose image is simple to convert, but extremely difficult to invert given only the image of a random input; and trapdoor information - a piece of information that allows the inverse to be calculated with ease. Before understanding the one-way-function that Diffie and Hellman selected, one must first understand the Discrete Logarithm Problem.

2 The Discrete Logarithm Problem

Diffie and Hellman’s first public key construction drew inspiration heavily from the Discrete Logarithm Problem in a finite field with prime order, $\mathbb{F}_p$. By the Primitive Root Theorem, there exists an element $g \in \mathbb{F}_p$, whose powers generate every element of $\mathbb{F}_p$, making $\mathbb{F}_p$ of the form:

$$\mathbb{F}_p^* = \{1, g, g^2, g^3, ..., g^{p-2}\}$$

The element g is called a primitive root or generator of $\mathbb{F}_p^*$.

Building on this idea, the Discrete Logarithm Problem (DLP) is the problem of finding an exponent, x, such that

$$g^x \equiv h \pmod{p}$$

for some nonzero element $h \in \mathbb{F}_p$. Here, x is called the discrete logarithm of h to the base g, denoted $\log_g(h)$. The DLP can be extended to any arbitrary group G under that group’s group laws and it turns out to be extremely computationally expensive and time consuming to crack when instantiated with a sufficiently large value of p. In fact, in the context of cryptography, security is only reached when it would take a computer longer than the age of the universe to solve this problem.

3 The Diffie-Hellman Key Exchange

One of the biggest issues when trying to communicate across an insecure channel is both parties agreeing on a key to use for their encryption, without the key falling into the hands of an adversary. While this task may appear impossible at first, Diffie and Hellman discovered a way to leverage the Discrete Logarithm Problem in a way that would allow two parties to securely share a key over an insecure channel. The Key Exchange Protocol between two parties, Alice and Bob, is as follows:

1. Alice and Bob agree on a large prime value, p, and a nonzero integer, g modulo p. These values are then published publicly.

2. Now, Alice and Bob independently pick secret integers a, and b, respectively.

3. Alice calculates $A \equiv g^a \pmod{p}$ while Bob calculates $B \equiv g^b \pmod{p}$.

4. Next, Alice and Bob send their computed values A and B to one another over any communication channel.

5. Finally, Alice uses Bob's computed value to calculate $A' \equiv B^a \pmod{p}$, while Bob uses Alice's computed value to calculate $B' \equiv A^b \pmod{p}$. Now, Alice and Bob can utilize the value of A' and B' as their private key for their encryption protocol.

Remark: The second set of values that Alice and Bob independently compute, A' and B', are equivalent due to the following:

$$A' \equiv B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \equiv B' \pmod{p}$$

3.1 Example of the Diffie-Hellman Key Exchange

Alice and Bob communicate over an insecure channel and agree upon prime number $p = 2011$ and primitive root $g = 986$. Alice selects her personal secret $a = 231$ and Bob selects his personal secret $b = 536$. Alice computes $A = 1727 \equiv 986^{231} \pmod{2011}$ while Bob computes $B = 1162 \equiv 986^{536} \pmod{2011}$. Both parties then exchange their values A and B over an insecure communication channel, holding on to their personal secrets a, and b. Now Alice and Bob compute their shared secret:

$$1948 \equiv 986^{231 \cdot 536} \equiv A^b \equiv B^a \pmod{2011}$$

Consider a third party, Charlie, who eavesdropped on Alice and Bob's communication channel. This means that Charlie has access to the values $p = 2011$, $g = 986$, $A = 1727$, and $B = 1162$. If Alice and Bob agreed to use their shared secret as the key for an encryption protocol for further communication on the channel, Charlie would need to use the values he knows to find either Alice's personal secret a, or Bob's personal secret b. To do this, Charlie would need to solve either of the following:

$$986^a \equiv 1727 \pmod{2011} \qquad \text{or} \qquad 986^b \equiv 1162 \pmod{2011}$$

Both of these problems take the form of the Discrete Log Problem. This example illustrates the difficulty Charlie would have finding Alice and Bob's shared secret as she would need to check all possible powers of 986 mod 2011 in order to discover a and b. Due to the small magnitude of the numbers in this example problem, Charlie could brute force the problem and sequentially calculate each power; however, in real world applications, Alice and Bob would agree on a much larger p value. In fact, current guidelines suggest selecting a p value with 1000 bits and picking g approximately equal to $p/2$ with prime order. Following these guidelines makes the DLP entirely impractical to solve without the application of a quantum computer.

4 The Diffie Hellman Problem

Stemming from this example, an eavesdropper like Charlie has access to the values of g, p, A, B, g^a, and g^b, and we've seen that deriving the values of a, and b can prove to be quite difficult. However, this does not necessarily guarantee the security of Alice and Bob's communication using an encryption protocol with their shared secret. This raises the following question about the security of the Diffie-Hellman Problem (DHP): Let p be a prime and g an integer. Given known values $g^a \pmod{p}$ and $g^b \pmod{p}$, how can one compute $g^{ab} \pmod{p}$ using only what is known? Clearly, if Charlie can solve the DLP she can gain access to either of Alice or Bob's private keys. However, it is unknown whether solving the DHP makes it easier to solve the DLP.

5 Computational Difficulties

One of the practical difficulties of implementing the Diffie-Hellman Key Exchange is that the numbers involved grow very quickly. As mentioned above, a real world implementation would require a g value of about 500 bits. This is equivalent to raising a several hundred (decimal) digit number to a very large power before reducing modulo p. This intermediate value can grow to be larger than the number of particles in the universe. This begs the question of how computers can implement the algorithm without running out of memory or time. The solution lies in a procedure known as the *Binary Exponential Algorithm* which successively squares the base g before reducing modulo p at every step. The specific procedure is outlined as follows:

1. Write out the exponent a in its binary form.
2. Calculate the necessary powers of q^{2^j} where j is the number of bits needed to represent a in binary minus one. Reduce modulo p after every step.
3. Multiply together the results that correspond to a 1 in the binary representation of a. Reduce mod p whenever the numbers grow too large.

5.1 Example of the Binary Exponential Algorithm

Suppose we want to compute $141^{47} \pmod{1537}$. The intermediate value of 141^{47} is already a 102-digit number. Keep in mind that this is nowhere near the magnitude of the numbers involved in a real-world applicaiton of Diffie-Hellman. We first express the exponent 47 in binary form as

$$47 = 32 + 8 + 4 + 2 + 1 = (101111)_2$$

Next, we need to obtain the powers of $141^{2^j} \pmod{1537}$ for $0 \leq j \leq 5$ (Since 101111 is a 6 bit number) by repeatedly squaring and reducing modulo 1537 after each result:

$$\begin{aligned} 141^2 &\equiv 1437 \pmod{1537} & 141^{16} &\equiv 364 \pmod{1537} \\ 141^4 &\equiv 778 \pmod{1537} & 141^{32} &\equiv 314 \pmod{1537} \\ 141^8 &\equiv 1243 \pmod{1537} \end{aligned}$$

We now multiply together the powers of 141 that have a 1 in the binary representation of 47 and get that

$$\begin{aligned} 141^{47} &= 141^{32+8+4+2+1} \\ &= 141^{32} \cdot 141^8 \cdot 141^4 \cdot 141^2 \cdot 141^1 \\ &\equiv 314 \cdot 1243 \cdot 778 \cdot 1437 \cdot 141 \equiv 658 \pmod{1537} \end{aligned}$$

Note that this doesn't require computing the 102-digit-long intermediate value. That being said, most computers can only handle integers up to 64 bits. This means that even with this trick, most implementations of the Diffie-Hellman Key Exchange require the use of Big Integer Libraries such as GMP.

6 References

1. An Introduction to Mathematical Cryptography, by Jeffrey Hoffstein et al., Springer, 2014, pp. 61–69.

2. Elementary Number Theory, by David M. Burton, McGraw-Hill, 2011, pp. 70-71.