1 The Diffie-Hellman Key Exchange System

1.1 Mechanism

Goal: Alice and Bob want to exchange a shared "secret key" via an insecure channel. (Incidentally, the shared secret key is usually used for encryption and decryption with a symmetric cipher.)

- Alice and Bob choose and publish a large prime p and an integer g with large prime order in $\mathbb{F}_p^*$.
- Alice chooses a secret $a \in \mathbb{Z}$, computes $A := g^a \mod p$, and sends $A \in \mathbb{F}_p^*$ to Bob via the insecure channel.
- Bob chooses a secret $b \in \mathbb{Z}$, computes $B := g^b \mod p$, and sends $B \in \mathbb{F}_p^*$ to Alice via the insecure channel.
- Bob computes $A^b = (g^a)^b = g^{ab} \in \mathbb{F}_p^*$.
- Alice computes $B^a = (g^b)^a = g^{ab} \in \mathbb{F}_p^*$.
- Alice and Bob have both arrived at $g^{ab} \in \mathbb{F}_p^*$, which is to be their shared secret.

1.2 Security of the Diffie-Hellman key exchange system

- The presumed security of the Diffie-Hellman key exchange system is based on the presumed difficulty in solving the

  The Diffie-Hellman Problem:
  Let p be a prime, and $g \in \mathbb{F}_p^*$. Given $A, B \in \langle\, g \,\rangle \subset \mathbb{F}_p^*$, find $g^{ab} \in \mathbb{F}_p^*$, where $a, b \in \mathbb{Z}$ are determined by $A = g^a$ and $B = g^b$.

  Note that the solution of the Diffie-Hellman problem does NOT require the knowledge of a and b, but only that of $g^{ab} \in \langle\, g \,\rangle$.

- It is clear that an efficient algorithm for the Discrete Logarithm Problem will lead to an efficient algorithm for the Diffie-Hellman Problem. An efficient algorithm of the Discrete Logarithm Problem will break the Diffie-Hellman key exchange system.

  The Discrete Logarithm Problem:
  Let p be a prime, and $g \in \mathbb{F}_p$ be a primitive element, i.e. $\mathbb{F}_p^* = \langle\, g \,\rangle$. Given any $h \in \mathbb{F}_p^*$, find $x \in \mathbb{Z}$ such that $h = g^x$.

2 The ElGamal Public Key Cryptosystem

2.1 Mechanism

Goal: Bob wants to send Alice an encrypted message via an insecure channel.

- Alice and Bob choose and publish a large prime p and an integer g with large prime order in $\mathbb{F}_p^*$.
- Alice chooses a secret $a \in \mathbb{Z}$, computes $A := g^a \mod p$, and sends $A \in \mathbb{F}_p^*$ to Bob via the insecure channel.
- Bob chooses a secret $b \in \mathbb{Z}$, computes $B := g^b \mod p$, and sends $B \in \mathbb{F}_p^*$ to Alice via the insecure channel.
- Bob computes $A^b = (g^a)^b = g^{ab} \in \mathbb{F}_p^*$.
- Alice computes $B^a = (g^b)^a = g^{ab} \in \mathbb{F}_p^*$.
- Alice and Bob have both arrived at $g^{ab} \in \mathbb{F}_p^*$, which is to be their shared secret.

2.2 Security of the Diffie-Hellman key exchange system

- The presumed security of the Diffie-Hellman key exchange system is based on the presumed difficulty in solving the

  The Diffie-Hellman Problem:
  Let p be a prime, and $g \in \mathbb{F}_p^*$. Given $A, B \in \langle\, g \,\rangle \subset \mathbb{F}_p^*$, find $g^{ab} \in \mathbb{F}_p^*$, where $a, b \in \mathbb{Z}$ are determined by $A = g^a$ and $B = g^b$.

  Note that the solution of the Diffie-Hellman problem does NOT require the knowledge of a and b, but only that of $g^{ab} \in \langle\, g \,\rangle$.

- It is clear that an efficient algorithm for the Discrete Logarithm Problem will lead to an efficient algorithm for the Diffie-Hellman Problem. An efficient algorithm of the Discrete Logarithm Problem will break the Diffie-Hellman key exchange system.

  The Discrete Logarithm Problem:
  Let p be a prime, and $g \in \mathbb{F}_p$ be a primitive element, i.e. $\mathbb{F}_p^* = \langle\, g \,\rangle$. Given any $h \in \mathbb{F}_p^*$, find $x \in \mathbb{Z}$ such that $h = g^x$.

3 The RSA Public Key Cryptosystem

3.1 Mechanism

Goal: Alice wants to send Bob an encrypted message through an insecure channel.

- Bob chooses his public key $(n, e) \in \mathbb{N}^2$ and private key $d \in \mathbb{N}$. Bob publishes his public key.
  - $n \in \mathbb{N}$ is called the modulus, with $n = pq$, where p and q are large distinct prime numbers. Note that Bob publishes n but keeps p and q secret.
  - $e \in \mathbb{N}$ is the called encryption exponent, and satisfies $\gcd(e, (p-1)(q-1)) = 1$.
  - $d \in \mathbb{N}$ is the called decryption exponent, and is determined by e and $n = pq$ via $d = e^{-1} \in \mathbb{Z}_{(p-1)(q-1)}$. Note that $e^{-1} \in \mathbb{Z}_{(p-1)(q-1)}$ exists since $\gcd(e, (p-1)(q-1)) = 1$.
- Alice
  - chooses plaintext $m \in \mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$.
  - encrypts her plaintext m using Bob's public key (n, e) by raising $m \in \mathbb{Z}_n$ to the e^{th} power. In other words, Alice computes her ciphertext $c = m^e \in \mathbb{Z}_n$.
  - sends to Bob through the insecure channel the ciphertext $c \in \mathbb{Z}_n$.
- Bob decrypts the ciphertext $c \in \mathbb{Z}_n$ from Alice by taking the e^{th} root of c in $\mathbb{Z}_n$ using his private key $d \in \mathbb{N}$ as follows:
$$c^d = (m^e)^d = m^{ed} = m^{1+k(p-1)(q-1)} = m \cdot (m^{(p-1)(q-1)})^k = m \cdot (1)^k = m \in \mathbb{Z}_n$$
  - The second last equality follows from $m^{(p-1)(q-1)} \equiv 1 \mod n$, which follows immediately from Euler's Theorem. It can also be justified with Fermat's Little Theorem as follows:
$$\text{Fermat's Little Theorem} \quad \Longrightarrow \quad \left\{ \begin{array}{ll} m^{(p-1)(q-1)} = \left(m^{p-1}\right)^{q-1} \equiv 1 \mod p, & \text{and} \\ m^{(p-1)(q-1)} = \left(m^{q-1}\right)^{p-1} \equiv 1 \mod q. & \end{array} \right.$$

    Hence, $m^{(p-1)(q-1)} - 1$ is divisble by both p and q, and hence also by $pq = n$ (since p and q are distinct primes). Thus, $m^{(p-1)(q-1)} \equiv 1 \mod n = pq$.

3.2 Comments

- One-way function (easy): Exponentiation in $\mathbb{Z}_n$.
  - Repeating Squaring Algorithm
- (Difficult) inverse function: Taking roots in $\mathbb{Z}_n$, for $n = pq$, where p and q are large distinct prime numbers.
- Trapdoor: If the factorization of $n = pq$ is known, then we can convert the inverse function (taking roots in $\mathbb{Z}_n$, which is slow) to an exponentiation in $\mathbb{Z}_n$, which is fast.

3.3 How to find large prime numbers?

- Generate a large N-bit (say $N = 1024$) random number x, i.e. $2^{N-1} < x < 2^N$. Use an efficient primality test to check whether x is prime. If so, we are done. If not, repeat until we succeed.
- The Prime Number Theorem (from Analytic Number Theory) gives an estimate of how many times we need to try before succeeding. The Prime Number Theorem states that

$$\lim_{x\to\infty} \frac{\pi(x)/x}{1/\ln(x)} \;=\; 1,$$

where $\pi(x)$ is the number of prime numbers less than or equal to x. Hence, it implies that, for large values of N, the probability that a randomly selected integer $x \in (2^{N-1}, 2^N)$ is prime is approximately

$$\frac{1}{\ln(2^N)}$$

Conversely, this implies that, on average, out of every $\ln(2^N) = N \cdot \ln(2) \approx 0.693 \cdot N$ randomly and independently selected integers from $(2^{N-1}, 2^N)$, one of them will be a prime number. For example, if $N = 1024$, then $0.693 \cdot N \approx 709.78$; in other words, if we are selecting random integers from $(2^{1023}, 2^{1024})$, then on average, we expect repeating approximately 710 times before we succeed in selecting a prime number. Note that $2^{1023} = 10^{1023\times\log_{10}(2)} \approx 10^{1023\times 0.301} \approx 10^{307.95}$.

- The Miller-Rabin Primality test
  - **Proposition** Let p be an odd prime and write $p - 1 = 2^k t$, where t is odd. Then, for each $a \in \mathbb{Z}$ with $p \nmid a$, one of the following is true:
    - $a^t \equiv 1 \mod p$, or
    - One of a^t, a^{2t}, a^{4t}, …, $a^{2^{k-1}t}$ is congruent to $-1 \mod p$.
  - **Corollary** Let $n \in \mathbb{Z}$ be an odd number, with $n - 1 = 2^k t$, t being odd. Then, n is composite, if any of the following is true:
    - There exists $a \in \mathbb{Z}$ such that $\gcd(a, n) > 1$.
    - There exists $a \in \mathbb{Z}$ such that $\gcd(a, n) = 1$, and $a^t \not\equiv 1 \mod n$, and $a^{2^i t} \not\equiv -1 \mod n$, for each $i = 0, 1, 2, \ldots, k - 1$.
  - **Proposition** Let n be an odd composite number. Then, at least 75% of integers between 1 and $n - 1$ are Miller-Rabin witnesses for n.

3.4 Factorization algorithms

- Pollard's $p - 1$ factorization algorithm

  This method "probably" works for producing a non-trivial factor for composite $n \in \mathbb{N}$ admitting a prime factor p such that $p - 1$ is a product of small primes.

  - **Proposition:** Let $n = pq$, where p and q are distinct prime numbers. Then the following two statements hold:

- For each $L \in \mathbb{N}$, we have the following implications:

$$(p-1) \mid L \quad \Longrightarrow \quad p \mid (a^L - 1)$$

- For any $a \in \mathbb{N}$ with $p \nmid a$ and $q \nmid a$, and any $L \in \mathbb{N}$,

$$\left.\begin{array}{l} p \mid (a^L - 1) \\ q \nmid (a^L - 1) \end{array}\right\} \quad \Longrightarrow \quad p = \gcd(a^L - 1, n)$$

– **Key observations:**
  - If $p-1$ is a product of small primes, then $p \mid N!$, for some not-too-large N.
  - If $(q-1) \nmid N!$, then $q \nmid (a^{N!} - 1)$ is "probably" true.
  - If $p-1$ is a product of small primes, and $q-1$ is NOT so, then computing $\gcd(a^{k!} - 1, n)$, for $k = 2, 3, \ldots$, will "probably" yield p as a non-trivial factor of n.

- Factorization via difference of squares

  – **Key observations:**

  Suppose we know that $n \in \mathbb{N}$ is odd and composite. We want to find a non-trivial factor of n.
  - If we can find $a, b \in \mathbb{N}$ such that n is the difference of their squares, i.e. $n = a^2 - b^2 = (a-b)(a+b)$, then computing $\gcd(a-b, n)$ will yield a non-trivial factor of n.
  - Conversely, suppose $n = cd$. Since n is odd, both c and d must also be odd. Hence, $a := \frac{1}{2}(c+d) \in \mathbb{Z}$ and $b := \frac{1}{2}(c-d) \in \mathbb{Z}$. And, $a^2 - b^2 = \cdots = cd = n$. In other words, every composite odd integer can be written as the difference of two squares.
  - If some multiple kn is a difference of squares, i.e. $kn = a^2 - b^2 = (a-b)(a+b)$, then computing $\gcd(a-b, n)$ will "probably" yield a non-trivial factor of n, since it should be unlikely that n divides $a - b$.
  - In summary, if we could find $a, b \in \mathbb{Z}$ such that $a^2 \equiv b^2 \mod n$, then computing $\gcd(a-b, n)$ will probably yield a non-trivial factor of n.

  – **Outline of general procedure:**
  1. **Find B-smooth perfect squares in $\mathbb{Z}_n$.** Find many $a_1, a_2, \ldots, a_r \in \mathbb{Z}$ such that every prime factor of $c_i \equiv a_i^2 \mod n$ is less than or equal to B.
  2. Find sub-collections $c_{i_1}, c_{i_2}, \ldots, c_{i_s}$ such that $c_{i_1} c_{i_2} \cdots c_{i_s} \equiv b^2 \mod n$ are perfect squares in $\mathbb{Z}_n$.
  3. Let $a := a_{i_1} a_{i_2} \cdots a_{i_s} \mod n$. Then, computing $\gcd(a-b, n)$ will probably yield a non-trivial factor of n.

  – Comments on the general procedure:
  - Step (3) can be performed efficiently using the Euclidean Algorithm.
  - Step (2) is equivalent to solving a homogeneous (sparse) system of linear equations over $\mathbb{F}_2$.
  - The main challenge in difference-of-squares factorization is Step (1), namely, given $n \in \mathbb{Z}$, finding enough B-smooth perfect squares in $\mathbb{Z}_n$.