**Задание №1: Алгоритм RSA**

RSA (Rivest-Shamir-Adleman) - это асимметричный криптографический алгоритм, который использует пары ключей: открытый и закрытый. Процесс работы RSA включает следующие шаги:

**1. Генерация ключей:**

- Генерация двух больших простых чисел, обычно обозначаемых как `p` и `q`.

- Вычисление их произведения `n = p \* q`. `n` используется как модуль для обоих ключей.

**2. Выбор открытой экспоненты:**

- Выбор числа `e` (открытой экспоненты) такого, что `1 < e < φ(n)`, где `φ` - функция Эйлера от `n`.

- `e` должно быть взаимно простым с `φ(n)` (не иметь общих делителей, кроме 1).

**3. Вычисление закрытой экспоненты:**

- Нахождение числа `d` (закрытой экспоненты) такого, что `(d \* e) mod φ(n) = 1`.

**4. Формирование открытого и закрытого ключей:**

- Открытый ключ: `{e, n}`

- Закрытый ключ: `{d, n}`

**5. Шифрование:**

- Преобразование сообщения `m` в число `M`, где `0 <= M < n`.

- Шифрование с использованием открытого ключа: `c = (M^e) mod n`.

**6. Дешифрование:**

- Дешифрование с использованием закрытого ключа: `M = (c^d) mod n`.

- Преобразование числа `M` обратно в сообщение.

**Задание №2: Алгоритм Диффи-Хеллмана**

Алгоритм Диффи-Хеллмана используется для безопасного обмена секретными ключами через непротиворечивый канал. Процесс включает следующие шаги:

**1. Выбор параметров:**

- Выбор большого простого числа `p`.

- Выбор первообразного корня по модулю `p`, обозначаемого как `g`.

**2. Генерация ключей:**

- Каждая сторона выбирает свое секретное значение `a` или `b`.

- Вычисление открытого ключа: `A = (g^a) mod p` или `B = (g^b) mod p`.

**3. Обмен открытыми ключами:**

- Стороны обмениваются вычисленными открытыми ключами `A` и `B`.

**4. Вычисление общего секрета:**

- Каждая сторона вычисляет общий секрет: `s = (B^a) mod p` или `s = (A^b) mod p`.

**5. Использование общего секрета:**

- Общий секрет `s` используется как секретный ключ для симметричного шифрования.

**Задание №3: Алгоритм Эль-Гамаля**

Алгоритм Эль-Гамаля также является асимметричным алгоритмом. Процесс работы включает следующие шаги:

**1. Генерация ключей:**

- Выбор большого простого числа `p`.

- Выбор первообразного корня по модулю `p`, обозначаемого как `g`.

- Выбор случайного секретного ключа `x` такого, что `1 < x < p-1`.

- Вычисление открытого ключа `y = (g^x) mod p`.

**2. Шифрование:**

- Выбор случайного числа `k` такого, что `1 < k < p-1` и `k` взаимно просто с `p-1`.

- Вычисление первой части шифртекста: `a = (g^k) mod p`.

- Вычисление второй части шифртекста: `b = (m \* (y^k)) mod p`, где `m` - сообщение.

3. Дешифрование:

- Вычисление обратной части шифртекста: `a^(-x) mod p`.

- Умножение этой обратной части на вторую часть шифртекста: `b \* a^(-x) mod p`.

### Задание №4\*: Использование криптографических библиотек

Для создания приложения с использованием криптографических библиотек и анализа работы алгоритмов RSA, Диффи-Хеллмана и Эль-Гамаля, вы можете воспользоваться библиотеками, такими как OpenSSL (для C/C++), Bouncy Castle (для Java и C#), Crypto++ (для C++), PyCryptoDome (для Python), или любой другой криптографической библиотекой, поддерживающей эти алгоритмы.

Пример создания приложения с использованием библиотеки Bouncy Castle на C#:

// Пример использования Bouncy Castle для RSA

using System;

using Org.BouncyCastle.Crypto;

using Org.BouncyCastle.Security;

using Org.BouncyCastle.Math;

using Org

.BouncyCastle.Crypto.Parameters;

class Program

{

static void Main()

{

// Генерация ключей RSA

var keyPairGenerator = GeneratorUtilities.GetKeyPairGenerator("RSA");

keyPairGenerator.Init(new KeyGenerationParameters(new SecureRandom(), 2048));

AsymmetricCipherKeyPair keyPair = keyPairGenerator.GenerateKeyPair();

// Пример шифрования и дешифрования сообщения

string сообщение = "Hello, RSA!";

byte[] зашифрованныйТекст = ШифроватьRSA(сообщение, keyPair.Public);

string расшифрованныйТекст = ДешифроватьRSA(зашифрованныйТекст, keyPair.Private);

Console.WriteLine("Исходное сообщение: " + сообщение);

Console.WriteLine("Зашифрованный текст: " + BitConverter.ToString(зашифрованныйТекст).Replace("-", ""));

Console.WriteLine("Расшифрованный текст: " + расшифрованныйТекст);

}

static byte[] ШифроватьRSA(string сообщение, AsymmetricKeyParameter открытыйКлюч)

{

var шифр = CipherUtilities.GetCipher("RSA/ECB/PKCS1Padding");

шифр.Init(true, открытыйКлюч);

byte[] байтыСообщения = System.Text.Encoding.UTF8.GetBytes(сообщение);

return шифр.DoFinal(байтыСообщения);

}

static string ДешифроватьRSA(byte[] зашифрованныйТекст, AsymmetricKeyParameter закрытыйКлюч)

{

var шифр = CipherUtilities.GetCipher("RSA/ECB/PKCS1Padding");

шифр.Init(false, закрытыйКлюч);

byte[] расшифрованныйТекст = шифр.DoFinal(зашифрованныйТекст);

return System.Text.Encoding.UTF8.GetString(расшифрованныйТекст);

}

}