

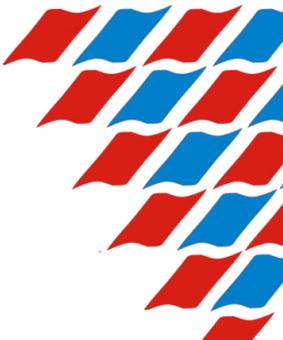
АСИММЕТРИЧНЫЕ АЛГОРИТМЫ

Дисциплина: Криптографические методы защиты информации
Преподаватель: Миронов Константин Валерьевич
Поток: БПС-3, ИКТ-5
Учебный год: 2020/21



Содержание лекции

- **Базовые операции асимметричной криптографии**
 - **Генерация случайных простых чисел**
 - **Операции по модулю целого числа**
 - **Поля, группы, подгруппы**
 - **Контроль правильности вычислений**
- Алгоритм Диффи-Хеллмана
- Алгоритм RSA
- Алгоритм DSA



Общие сведения

Асимметричная криптография

Основана на вычислительной сложности задач:

- **Дискретное логарифмирование** целых чисел: если $y = \alpha^x \bmod p$, где p – большое простое число, то вычислительно невозможно определить x , зная y , α и p
- **Факторизация больших чисел**: если $n = p * q$, где p и q – большие простые числа, то вычислительно невозможно определить p и q , зная n
- Дискретное логарифмирование в поле **эллиптической кривой**

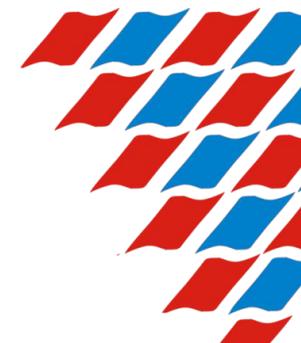
Особенности:

- Сложность может существенно снизиться при применении квантовых вычислений, т.е. **асимметричные алгоритмы потенциально уязвимы к квантовому криптоанализу**
- Время выполнения операций в асимметричной криптографии зависит от данных, т.е. **асимметричные алгоритмы потенциально уязвимы к тайминг-атакам**



Генерация случайных простых чисел

- Частота простых чисел уменьшается с порядком
 - В промежутке между числами $n/2$ и n вероятность, что случайное число будет простым, составляет примерно $1/\ln(n)$
 - Пример: вероятность, что случайное число длиной 2000 бит будет простым, составляет $\sim 1/1386$
 - В асимметричной криптографии используются простые числа длиной 1024-4096 бит (алгоритм RSA) или 2048-8192 бит (алгоритм Диффи-Хеллмана)
- Генерируются случайные нечетные числа в нужном диапазоне и проверяются на простоту
 - При генерации в старший и младший биты записываются единицы, в остальные биты записываются случайные данные
- Проверка случайного числа на простоту:
 - Проверяется, делится ли оно на числа из **списка малых простых чисел**
 - Если нет, то число проверяется с помощью **теста Рабина-Миллера**



Генерация случайных простых чисел

Список малых простых чисел

- Содержит все простые числа от 2 до некоторого значения
- С вычислительной точки зрения нецелесообразно составлять список с максимальным значением, большим чем 2^{20}
- Можно взять готовый список, либо быстро составить его при помощи алгоритма

Решето Эратосфена

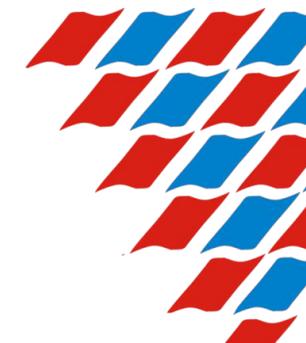
шаг 1. в список заносятся все числа от 1 до \max

шаг 2. для каждого i в списке

для j от 2 до $[\max/i]$

ЕСЛИ $i*j$ в списке ТО исключить $i*j$ из списка

- В результате все составные числа исключаются, все простые - остаются



Генерация случайных простых чисел

Тест Рабина-Миллера

- Вначале генерируются два числа t и s , удовлетворяющие условиям:
 - s - нечетное
 - $2^t s = p - 1$ где p - предполагаемое простое число
- Алгоритм генерации s и t :

$s = p - 1;$

$t = 0;$

ПОКА $s \bmod 2 = 0$

$s = s / 2;$

$t = t + 1;$

КОНЕЦ

Генерация случайных простых чисел

Тест Рабина-Миллера

- Выбирается случайное число a (**базис**) от 2 до p и выполняется проверка:

$i = 0$;

$v = a^s \bmod p$; // наиболее ресурсозатратная операция

ЕСЛИ $v = 1$ ТО ПРОВЕРКА ПРОЙДЕНА

ИНАЧЕ ПОКА $v \neq t - 1$

 ЕСЛИ $i = t - 1$ ТО ПРОВЕРКА НЕ ПРОЙДЕНА

 ИНАЧЕ

$v = v^2 \bmod p$;

$i = i + 1$;

 КОНЕЦ

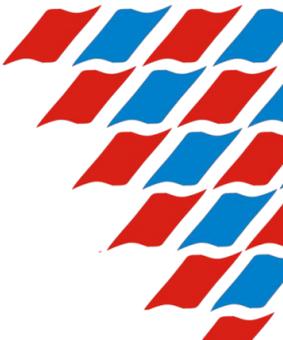
КОНЕЦ

 ПРОВЕРКА ПРОЙДЕНА

Простое p проходит проверку для всех базисов, составное – для 25%

Проверка проводится много раз с разными базисами

7 Рекомендуется проверка на 64 базисах, тогда вероятность ошибки 2^{-128}



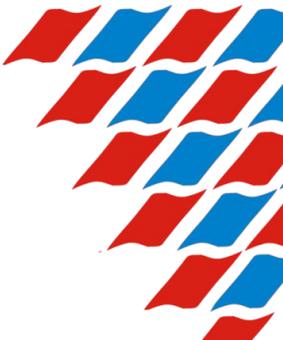
Операции по модулю целого числа

- Запись $8 \bmod 7 = 1$ равнозначна записи $8 = 1 \bmod 7$
- Если модуль является степенью 2, то при сложении/вычитании/умножении достаточно игнорировать разряды старше чем двоичный логарифм модуля
- Сложение/вычитание – достаточно выполнить операцию в целых числах, а затем (если нужно) один раз отнять или прибавить модуль

Операции по модулю целого числа

Умножение

- Сначала выполняется целочисленное умножение, затем вычисляется остаток от целочисленного деления результата на модуль [целочисленное деление ресурсозатратно]
- Ускоренный алгоритм для простого модуля – **умножение Монтгомери**
 - Функция $C = MontPro(A, B, p) = (A * B * 2^{-k}) \bmod p$ если выполняются условия:
 - p – нечетное число,
 - $A * B < (p - 1)^2$,
 - k – число двоичных разрядов в p
 - Как посчитать результат $C = (A * B) \bmod p$ с использованием функции $MontPro$:
 - $K = 2^{2k} \bmod p$ // здесь все же придется считать остаток, но только один раз для каждого модуля
 - $A^* = MontPro(A, K, p)$
 - $B^* = MontPro(B, K, p)$
 - $C^* = MontPro(A^*, B^*, p)$
 - $C = MontPro(C^*, 1, p)$



Операции по модулю целого числа

Умножение Монгомери

- Функция MontPro (версия с побитным сдвигом):

$$C = A * B$$

для i от 1 до k

$C = C + (C \bmod 2) * p$ // т.е. если C нечетное число, к нему прибавляется модуль

$C = C/2$ //деление в данном случае сводится к стиранию равного нулю правого бита

если $C > p$ то $C = C - p$

- Версия со сдвигом на w бит (обычно w определяется разрядностью процессора)

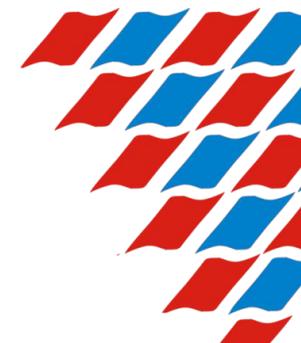
...

правые w бит C

$C = C + (C \bmod 2^w) * z * p$ //в результате правые w бит обнуляются

$C = C/2^w$ //деление сводится к стиранию обнуленных правых w бит

...



Операции по модулю целого числа

Деление по модулю

- При делении используется **расширенный алгоритм Евклида** для поиска **наибольшего общего делителя** двух чисел
- Для входных чисел a и b алгоритм возвращает 3 параметра: d (НОД a и b), u и v , удовлетворяющие условию $d = ua + vb$
- Результат деления вычисляется по формуле: **$a/b \bmod p = a*u \bmod p$**
- Расширенный алгоритм Евклида:

Выполнимо не всегда, но для простых модулей работает

```
a1 = a; b1 = b; u_a = 1; v_a = 0; u_b = 0; v_b = 1;
```

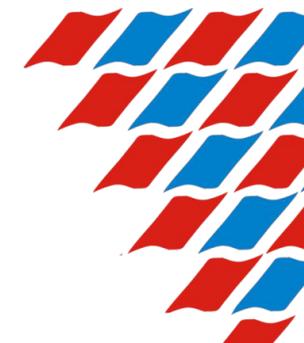
```
ПОКА a1 != 0
```

```
    q = b1 / a1;
```

```
    [ a1; b1 ] = [ b1-q*a1; a1 ];
```

```
    [ u_a; v_a; u_b; v_b ] = [ u_b-q*u_a; v_b-q*v_a; u_a; v_a ]
```

```
d = b1; u = u_b; v = v_b;
```



Операции по модулю целого числа

Возведение в натуральную степень по модулю простого числа

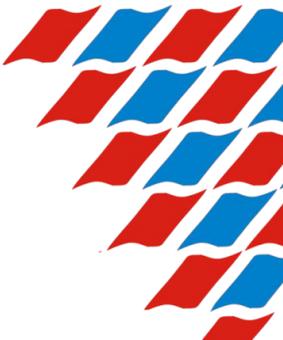
Функция Диффи-Хеллмана:

$$y = f_{DH}(x, \alpha, p) = \alpha^x \bmod p$$

Свойство 1. Функция односторонняя относительно x : нельзя вычислить x даже зная y , α и p . Задача такого вычисления называется **дискретным логарифмированием**

Свойство 2. Функция коммутативна относительно x :

$$\alpha^{x*z} \bmod p = (\alpha^x \bmod p)^z \bmod p = (\alpha^z \bmod p)^x \bmod p$$



Операции по модулю целого числа

Возведение в натуральную степень по модулю простого числа

Рекурсивный алгоритм вычисления $y = f_{DH}(x, \alpha, p) = \alpha^x \bmod p$

ЕСЛИ $x = 0$ ТО $y = 1$

ИНАЧЕ-ЕСЛИ $x = 1$ ТО $y = a$

ИНАЧЕ-ЕСЛИ $x = 2$ ТО $y = \alpha^2 \bmod p$

ИНАЧЕ-ЕСЛИ $x \bmod 2 = 0$ ТО

$$y_0 = \alpha^{\frac{x}{2}} \bmod p;$$

$$y = y_0^2 \bmod p;$$

ИНАЧЕ

$$y_0 = \alpha^{\frac{x-1}{2}} \bmod p;$$

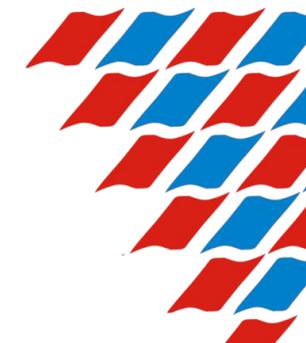
$$y = (\alpha * y_0^2) \bmod p;$$

Операции по модулю целого числа

Возведение в натуральную степень по модулю простого числа

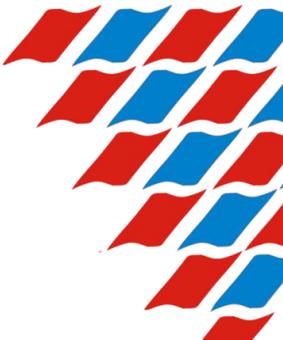
Особенности:

- Последовательность $1, \alpha, \alpha^2 \bmod p, \dots, \alpha^k \bmod p, \dots$ зацикливается после достижения значения $\alpha^q \bmod p = 1$. Здесь α называется **образующим элементом**, а q – **порядком**
- **Примитивный элемент** – такое значение α , что порядок равен $p-1$, а последовательность включает все числа от 1 до $p-1$
- Для любого α порядок является делителем числа $p-1$
- Любая такая последовательность является подгруппой мультипликативной группы $\{1, 2, \dots, p-1\}$ по модулю p



Поля, группы, подгруппы

- **Конечное поле по модулю p** – множество чисел $\{0, 1, 2, \dots, p-1\}$
- **Группа** – множество чисел, для которых определена некоторая операция, такая, что результатом ее выполнения над двумя некоторыми элементами, является некоторый третий элемент
 - **Аддитивная группа по модулю p** - $\{0, 1, 2, \dots, p-1\}$ и сложение по $\text{mod } p$
 - **Мультипликативная группа по модулю p** - $\{1, 2, \dots, p-1\}$ и умножение по $\text{mod } p$
- **Подгруппа** – группа, все члены которой также являются членами большей группы с той же операцией
 - Пример: $\{2, 4, 6\}$ – подгруппа мультипликативной группы по $\text{mod } 8$
 - Последовательность $1, \alpha, \alpha^2 \text{ mod } p, \dots, \alpha^{q-1} \text{ mod } p$ является подгруппой мультипликативной группы по $\text{mod } p$, где p – простое число и $\alpha^q \text{ mod } p = 1$
 - Порядок последовательности равен количеству элементов в наименьшей подгруппе такого типа, к которой относится образующий элемент

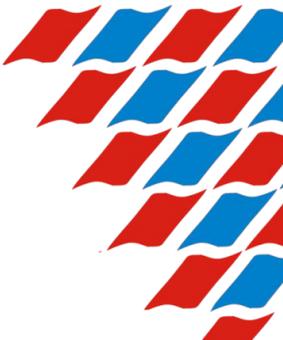


Контроль правильности вычислений

- Специализированные библиотеки позволяют относительно быстро производить арифметические операции над большими числами
- Как правило содержат низкоуровневый код, оптимизирующий вычисления на конкретной аппаратной платформе
- Ошибки в вычислениях могут привести к уязвимостям
- Проблема тестирования: работа кода зависит от самих данных, а разнообразие вариантов больше, чем можно протестировать

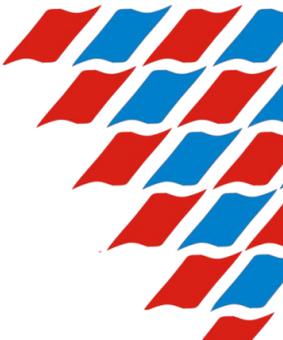
Подход: **вупинг**

- Вычисления дублируются по модулю случайного секретного простого числа t обычно длиной 32, 64 или 128 бит
- После каждой операции проверяется, что ее результат, взятый по $\text{mod } t$ равен результату дублированных вычислений
- Вероятность, что при вупинге не будет обнаружена случайная ошибка, примерно равна $1/t$



Содержание лекции

- Базовые операции асимметричной криптографии
- **Алгоритм Диффи-Хеллмана**
 - **Описание алгоритма**
 - **Выбор параметров**
- Алгоритм RSA
- Алгоритм DSA



Общие сведения

Асимметричная криптография

Согласование
секретных
ключей

Алгоритм Диффи-Хеллмана

Асимметрично
е шифрование

RSA

Алгоритм Эль-Гамала

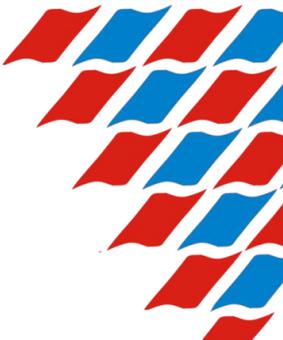
Алгоритм Рабина

Электронна
я подпись

DSA

ГОСТ Р 34.10

Любой алгоритм
асимметричного
шифрования можно
применять для передачи
секретных ключей в
зашифрованном виде



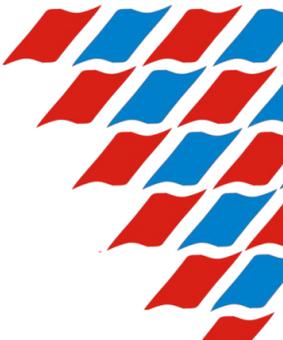
Алгоритм Диффи-Хеллмана

Используются операции в мультипликативной группе по $\text{mod } p$, где p – простое число длиной 2048-8192 бит

шаг 1. Алиса генерирует случайный ключ x_A и вычисляет $y_A = f_{DH}(x_A, \alpha, p)$. Боб генерирует случайный ключ x_B и вычисляет $y_B = f_{DH}(x_B, \alpha, p)$.

шаг 2. Алиса и Боб обмениваются y_A и y_B по открытому каналу.

шаг 3. Алиса и Боб вычисляют общий ключ шифрования $y_{AB} = f_{DH}(x_A, y_B, p) = f_{DH}(x_B, y_A, p)$.



Алгоритм Диффи-Хеллмана

- Полученный общий ключ затем преобразуется в ключ шифрования с помощью KDF
- Алгоритм может использоваться при количестве абонентов больше 2
 - «Цифровая телефонная книга» – для каждого i -го абонента указывается его y_i
 - Абонент j при необходимости установить связь с i вычисляет y_{ij}
- **Проблема:** если основание степени образует подгруппу с малым порядком, то множество возможных значений значений y_A или y_B может оказаться достаточно невелико, чтобы их можно было подобрать полным перебором
 - Необходим алгоритм выбора основания степени, дающего высокий порядок
 - Необходим алгоритм проверки основания степени на высокий порядок



Алгоритм Диффи-Хеллмана

Выбор параметров: подгруппа с заданным порядком

Метод позволяет генерировать $\{\alpha, p\}$ такие, что порядок q будет иметь заданную длину:

шаг 1. Сгенерировать случайное простое число q заданной длины

шаг 2. Сгенерировать случайное четное число N длиной $\text{length}(N) = \text{length}(p) - \text{length}(q)$

шаг 3. Проверить $p = Nq + 1$ на простоту

шаг 4. Повторять шаги 2 и 3 пока p не станет простым

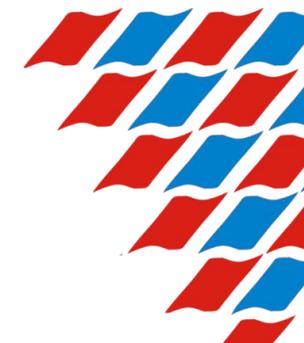
шаг 5. Сгенерировать случайное число $g \in \{1, \dots, p\}$

шаг 6. Рассчитать $\alpha = g^N \bmod p$

шаг 7. Проверить, что $\alpha^q \bmod p = 1$ и $\alpha \neq 1$

шаг 8. Повторять шаги 5-7 пока α не удовлетворит условиям

шаг 9. Вернуть $\{\alpha, p\}$



Алгоритм Диффи-Хеллмана

Выбор параметров: подгруппа с заданным порядком

- Длины q и N подбираются по следующему принципу
 - q равно требуемому количеству элементов в подгруппе
 - q длиной 256 бит дает примерно ту же безопасность, что 256-битный ключ симметричного шифрования
 - Длина N подбирается так, чтобы длина $p=Nq+1$ была 2048-8192 бит
- Метод позволяет упростить вычисление функции Диффи-Хеллмана благодаря свойству:
$$\alpha^e \bmod p = \alpha^{e \bmod q} \bmod p$$
- Помимо p и α общедоступной информацией должно быть значение q

Алгоритм Диффи-Хеллмана

Выбор параметров: подгруппа с заданным порядком

Проверка параметров алгоритма:

- Проверка $\{\alpha, p, q\}$
 - Проверить на простоту числа p и q
 - Проверить, что $(p-1) \bmod q = 0$
 - Проверить, что $\alpha^q \bmod p = 1$ и $\alpha \neq 1$
- При получении от Алисы y_A Боб должен убедиться, что $y_A^q \bmod p = 1$
 - Операции $y_A^q \bmod p$ и $y_A^{X_b} \bmod p$ можно выполнить одновременно с использованием **эвристики аддитивной последовательности** [additive sequence heuristics], что позволяет сэкономить ~ 250 операций умножения при использовании 256-битной подгруппы и 2048-битного модуля
- Кроме того, все параметры нужно проверять на соответствие их величины условиям

Содержание лекции

- Базовые операции асимметричной криптографии
- Алгоритм Диффи-Хеллмана
- **Алгоритм RSA**
 - **Описание алгоритма**
 - **Особенности практической реализации**
 - **Распределение ключей с помощью RSA**
- Алгоритм DSA

Алгоритм RSA

Генерация ключей

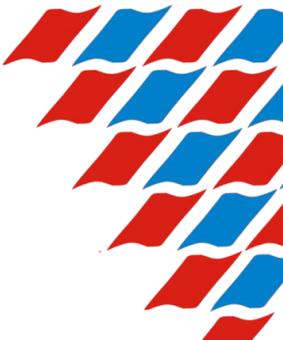
шаг 1. Сгенерировать простые числа p и q одинаковой заданной длины

шаг 2. Вычислить $n = pq$ и $t = \text{НОК}(p-1, q-1)$

шаг 3. Выбрать число e , взаимно простое с n

шаг 4. Вычислить $d = 1/e \bmod t$

- $\text{НОК}(p-1, q-1)$ вычисляется по формуле $\text{НОК}(a, b) = \frac{ab}{\text{НОД}(a, b)}$
- $\text{НОД}(a, b)$ вычисляется с помощью алгоритма Евклида
- Безопасной длиной p и q считается 1024-4096 бит
- В теории предполагается, что:
 - ОК = $\{e, n\}$
 - ЗК = $\{d\}$
 - p, q, t – секретные параметры, стираемые после генерации ЗК



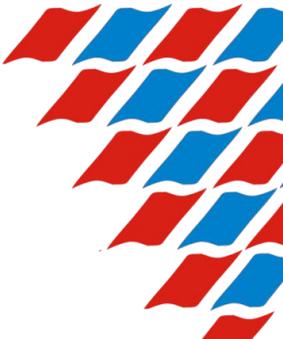
Алгоритм RSA

Зашифровка и расшифровка

$$\begin{aligned} \text{ШТ} &= \text{ОТ}^e \bmod n \\ \text{ОТ} &= \text{ШТ}^d \bmod n \end{aligned}$$

Постановка и проверка электронной подписи

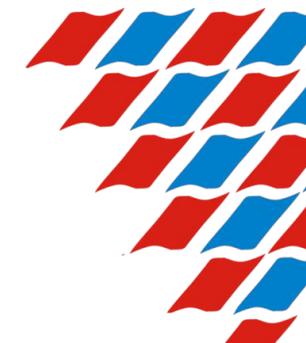
$$\begin{aligned} \text{ЭП} &= \text{ОТ}^d \bmod n \\ \text{ОТ} &= \text{ЭП}^e \bmod n \end{aligned}$$



Алгоритм RSA

Особенности практической реализации

- Значение e выбирается небольшим (часто $e=3$ для ЭП и $e=5$ для шифрования), чтобы снизить ресурсоемкость вычислений
 - Под ОК понимается n
 - Зашифрование и проверка ЭП намного менее ресурсозатратны, чем расшифрование и постановка ЭП
 - Разный показатель для шифрования и подписи позволяет шифровать текст, а затем его подписывать; иначе подписывание равнозначно расшифровке
- Зная n и любой из параметров $\{p, q, t, d\}$ можно рассчитать оставшиеся 3 параметра из $\{p, q, t, d\}$
- Расшифровку ШТ (постановку ЭП) можно ускорить примерно в 3-4 раза, зная p и q и используя **теорему Сунь Цзы** (китайскую теорему об остатках [Chinese Remainder Theorem, CRT])



Алгоритм RSA

Особенности практической реализации

Теорема Сунь Цзы [формулировка, используемая в RSA]: число x , взятое по модулю $n = p * q$ (где p и q – простые числа) взаимно-однозначно соответствует паре $\{a, b\}$ где $a = x \bmod p$ и $b = x \bmod q$

Формула Гарнера для вычисления x на основе $\{a, b\}$

$$x = \left(((a - b)(q^{-1} \bmod p)) \bmod p \right) q + b$$

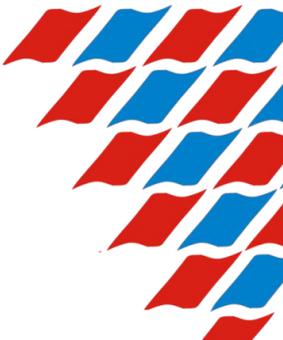
Пара $\{a, b\}$ является **CRT-представлением числа x**

$$x \leftrightarrow \{a, b\}$$

$$(x + y) \bmod n \leftrightarrow \{(a_x + a_y) \bmod p, (b_x + b_y) \bmod q\}$$

$$(xy) \bmod n \leftrightarrow \{(a_x a_y) \bmod p, (b_x b_y) \bmod q\}$$

$$(x^y) \bmod n \leftrightarrow \{(a_x^{y \bmod (p-1)}) \bmod p, (b_x^{y \bmod (q-1)}) \bmod q\}$$



Алгоритм RSA

Особенности практической реализации

Процедура генерации ключей с учетом практических соображений:

шаг 1. Сгенерировать нечетное число p заданной длины

шаг 2. Проверить, что $p \bmod 3 \neq 1$ и $p \bmod 5 \neq 1$

шаг 3. Проверить, что p - простое

шаг 4. Повторять шаги 1-3 пока p не пройдет обе проверки

шаг 5. Повторить шаги 1-4 для числа q

шаг 6. Вычислить $n = pq$ и $t = (p - 1)(q - 1) / \text{НОД}(p - 1, q - 1)$

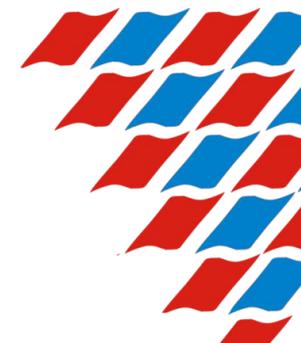
шаг 7. Вычислить $d_{\text{ЭП}} = 1/3 \bmod t$ и $d_{\text{Ш}} = 1/5 \bmod t$

$e_{\text{ЭП}} = 3, e_{\text{Ш}} = 5$

Алгоритм RSA

Особенности практической реализации

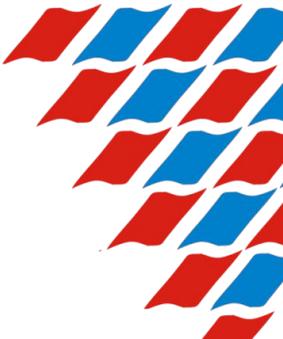
- Блок шифруемого текста должен иметь чуть меньшую длину, чем n
 - Если намного меньше – для расшифровки будет достаточно взять корень 5-й степени из ШТ
- Блок шифруемого текста перед шифрованием должен быть преобразован к псевдослучайному виду
 - Например – побитное сложение с солью
- При постановке ЭП сначала вычисляется хеш-функция от документа, затем ставится подпись на эту хеш-функцию
 - Чтобы достичь необходимой длины блока на основе хеш-функции можно сгенерировать ПСП
 - либо применить KDF, позволяющую варьировать длину выходного значения



Алгоритм RSA

Особенности практической реализации

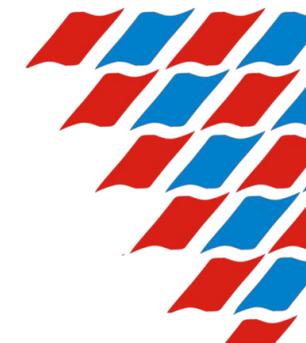
- Атака на *шифр* подбором открытого текста
 - Ева может проверить предположение о значении ОТ, зашифровав его и сравнив с имеющимся ШТ
 - То же относится и к другим асимметричным *шифрам*
- Проверка правильности вычислений
 - Вупинг
 - Проверка при постановке ЭП: после постановки провести процедуру проверки
 - Проверка при шифровании: сгенерировать случайное простое число z и проверить, что $(ШТ * z^e) \bmod n = (ОТ * z)^e \bmod n$



Алгоритм RSA

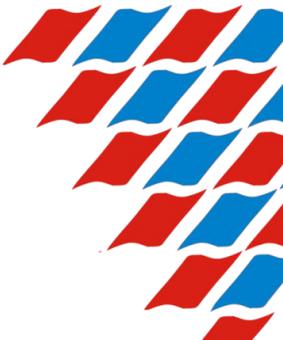
Использование для распределения ключей

- Асимметричное шифрование применяется для передачи ключа симметричного шифрования
- Алиса генерирует случайный блок ОТ, зашифровывает его с помощью RSA на ОК Боба и передает ШТ Бобу
- Боб расшифровывает блок своим ЗК
- Алиса и Боб генерируют ключ симметричного шифрования на основе переданного блока с помощью KDF
- Даже если Еве станет известен ключ симметричного шифрования, это не даст ей информации о ЗК Боба



Содержание лекции

- Базовые операции асимметричной криптографии
- Алгоритм Диффи-Хеллмана
- Алгоритм RSA
- **Алгоритм DSA**



Алгоритм DSA

- DSA (Digital Signature Algorithm) свободно распространяемый стандарт, АНБ, 1991
- DSS (Digital Signature Standard) – DSA на хеш-функцию SHA-1
- ECDSA – вариант на эллиптических кривых, 1999
- Сравнение с RSA
 - Не может применяться для шифрования
 - ЗК генерируется случайным образом и в целом генерация ключей проще
 - Постановка подписи по скорости аналогична RSA, проверка медленнее
- Параметры:

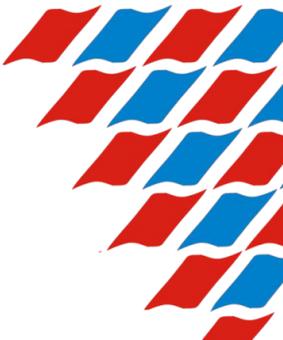
q – простое число той же длины, что и хеш-функция

p – простое число, величина кратна 64 битам, $(p - 1) \bmod q = 0$

a – число >1 , вычисляемое по формуле $a = h^{\frac{p-1}{q}} \bmod p$, где $h < p - 1$, обычно $h = 2$

ЗК – случайное число, меньше, чем q – длина обычно 256

ОК = $a^{\text{ЗК}} \bmod p$ – длина обычно 2048 бит



Алгоритм DSA

Процедура постановки:

k – секретное случайное число $< q$

$$r = (a^k \bmod p) \bmod q$$

$$s = \frac{\text{hash}(OT) + ZK * r}{k} \bmod q$$

$$\text{ЭП} = \{r, s\}$$

Процедура проверки:

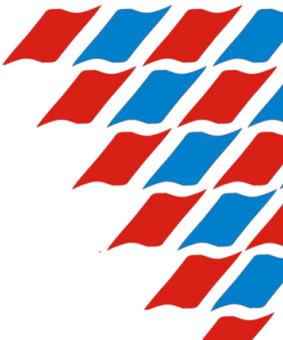
$$w = 1/s \bmod q$$

$$u_1 = (\text{hash}(OT) * w) \bmod q$$

$$u_1 = (r * w) \bmod q$$

$$v = (a^{u_1} * OK^{u_1} \bmod p) \bmod q$$

$$v =? r$$



Темы докладов

Универсальные асимметричные криптосистемы

- **Криптосистема Эль-Гамала**
- **Криптосистема Рабина**

Нормативно-правовое регулирование в области криптографии:

- **Федеральный Закон «Об электронной подписи»**
- **Порядок лицензирования средств криптографической защиты информации в РФ**
- **Нормативно-правовое регулирование криптографии в зарубежных странах**
(доклад можно поделить на троих; обязательно нужно осветить особенности регулирования в США, Китая и крупных стран ЕС)

