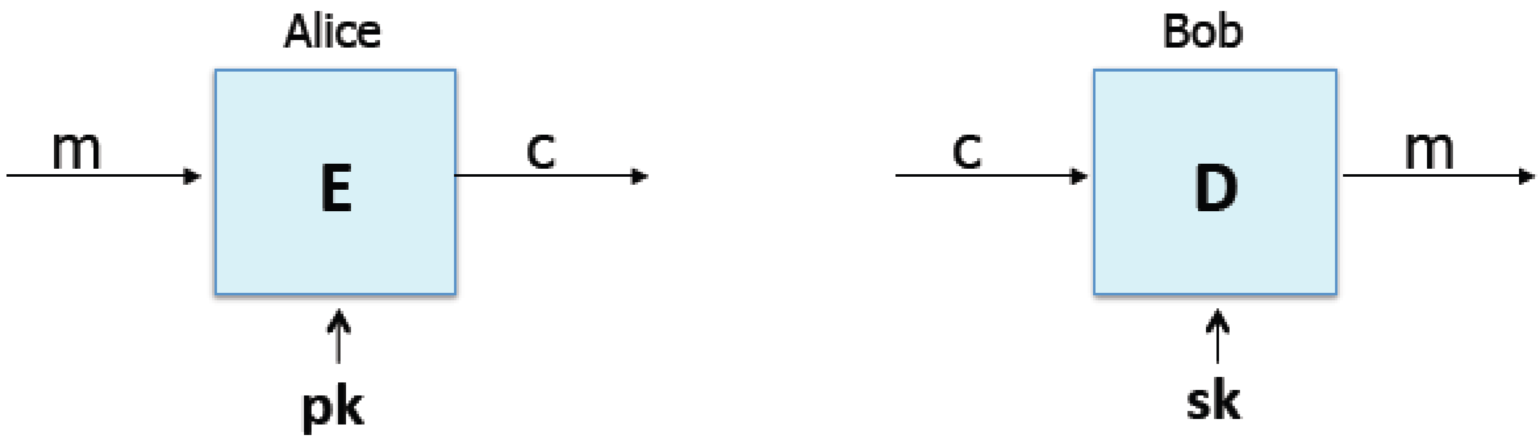
АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Перестановки с секретной дверью

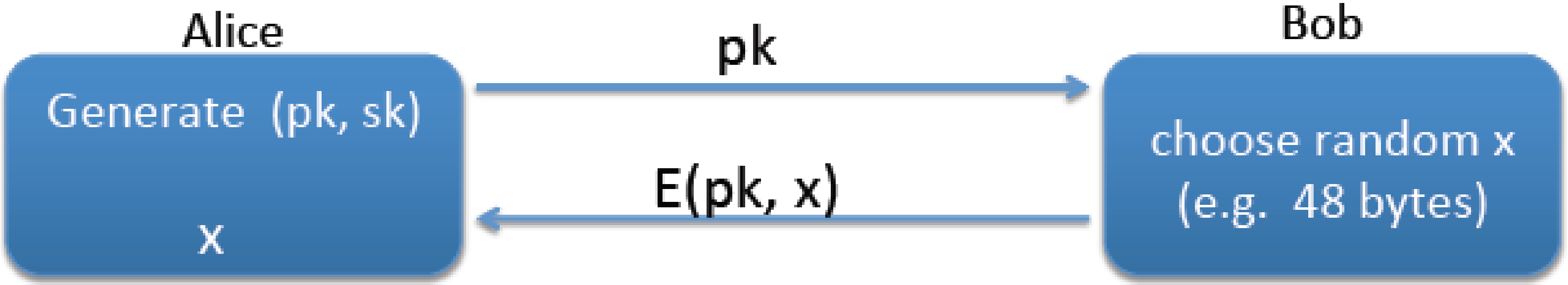
Шифрование с публичными ключами

- Боб формирует пару ключей (PK, SK) и передает публичный ключ Алисе



Приложения публичной криптографии

- **Установка сессии**



- **Не интерактивные приложения (почта):**
  - Боб отправляет сообщение Алисе на ее публичном ключе
  - Замечание: При этом Бобу необходимо получить публичный ключ Алисы (возникает задача управления ключей)

Шифрование на публичном ключе

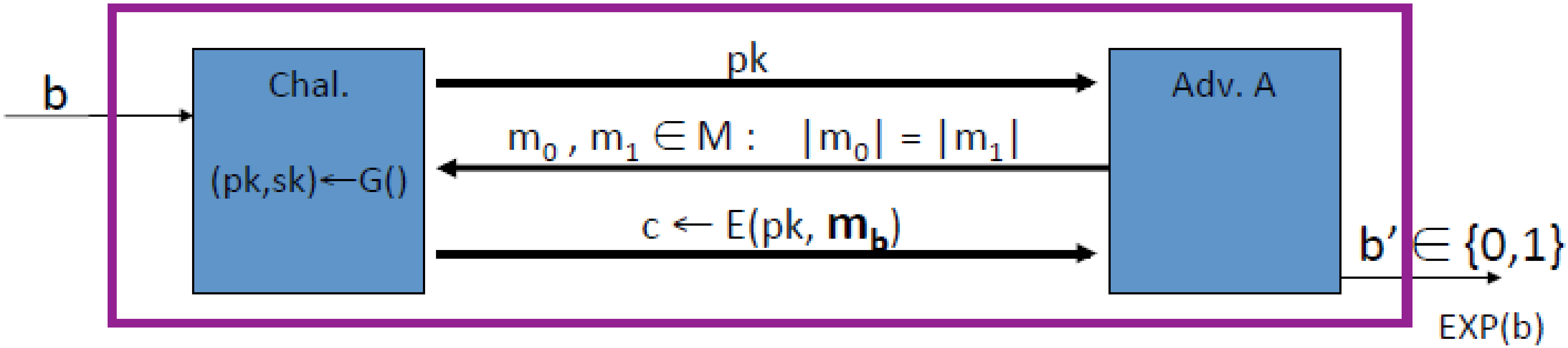
- Определение: Шифрование с публичным ключом – три алгоритма (G, E, D)
  - G(): рендомизированный алгоритм, который возвращает пару ключей (PK, SK)
  - E(PK, m): рендомизированный алгоритм, который принимает на вход $m \in M$ и возвращает $\mathrm{c} \in C$
  - D(SK, c): детерминированный алгоритм, который принимает на вход $\mathrm{c} \in C$ и возвращает $m \in M$ или $\perp$

- Состоятельность: $\forall (PK, SK)$ порожденные G:

$$\forall\ m \in M: \mathrm{D}\big(\mathrm{SK}, \mathrm{E}(\mathrm{PK}, \mathrm{m})\big) = \mathrm{m}$$

Безопасность: подслушивание

- Для b=0,1 определим эксперименты EXP(0) и EXP(1)



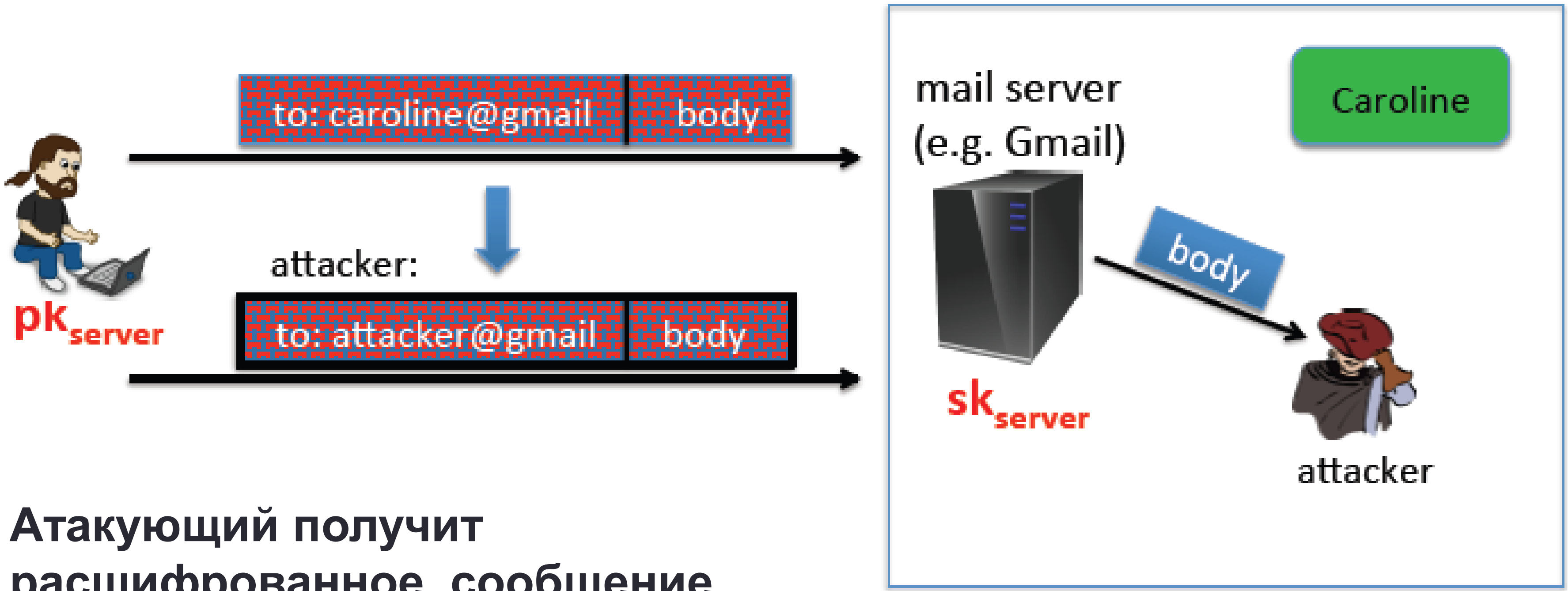
- Определение: **E**=(G, E, D) является сем. стойкой для любого атакующего A:
$$Adv_{SS}[\boldsymbol{E}, A] = |\Pr\{EXP(0) = 1\} - \Pr\{EXP(1) = 1\}| < neg$$

Сравнение с симметричными системами

- Симметричная криптография:
  - Стойкость: одноразовая и многоразовая
  - Из одноразовой стойкости не следует многоразовая
- Асимметричная криптография:
  - Из одноразовой стойкости следует многоразовая
  - Асимметричная криптография **должна** быть рандомизированная

Стойкость к активным атакам

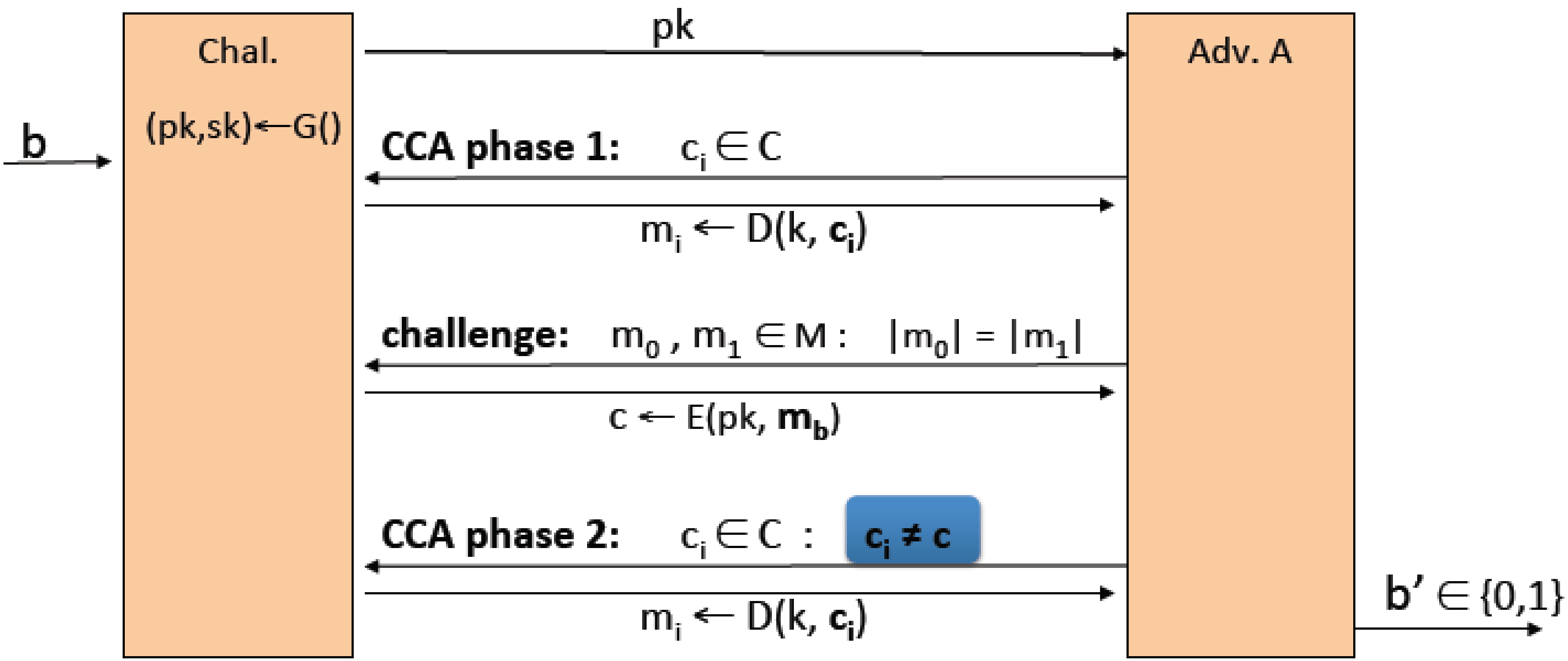
- Как можно подделать сообщение?



Атакующий получит расшифрованное сообщение, которое начинается to: attacker

Стойкость в модели CCA: случай асимметричтого шифрования

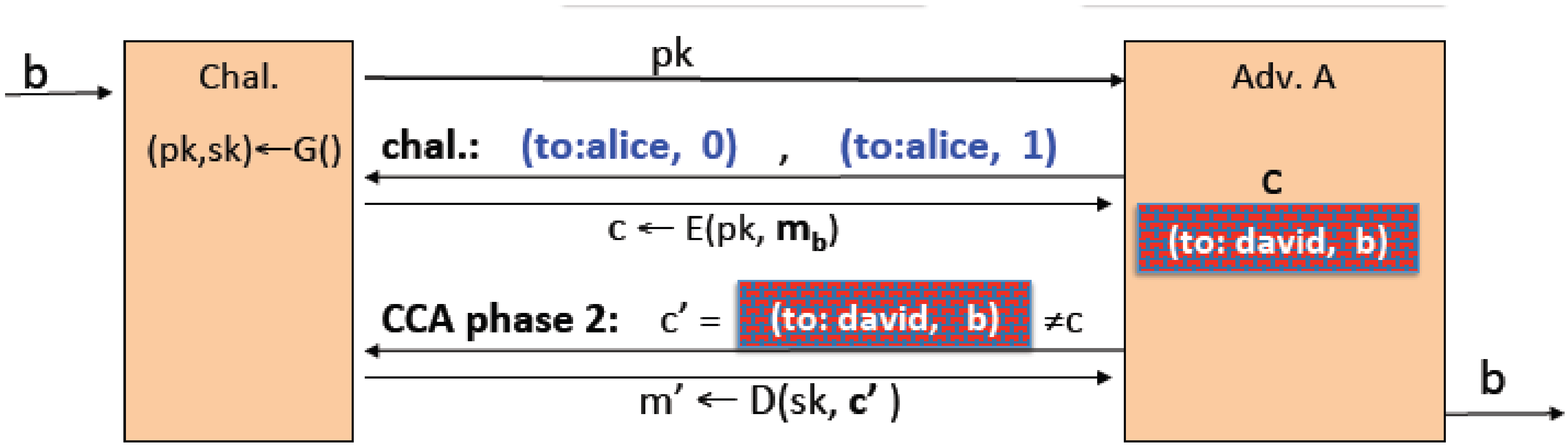
- Определим эксперимент EXP(b) для b=0,1 и асимметричной криптосистемы **E**=(G, E, D)



CCA

- Определение: **E**=(G, E, D) является CCA стойкой для любого атакующего A:
$Adv_{CCA}[\boldsymbol{E}, A] = |\Pr\{EXP(0) = 1\} - \Pr\{EXP(1) = 1\}| < neg$

- Пример: to:alice -> to:david



Активные атаки

- Симметричные шифросистемы:
  - Аутентифицированное шифрование
  - Атакующий не может создать новый шифротекст
  - Подразумевает CCA стойкость

- Асимметричные шифросистемы:
  - Атакующий **может** создать новый шифротекст при помощи PK
  - **Требует** CCA стойкость

Арифметика по модулю n

- $\mathbb{Z}_n^+$ — это группа по сложению.
- $\mathbb{Z}_n^*$ — это группа по умножению.
- Сколько элементов в $\mathbb{Z}_n^*$?
- Обратимые элементы в $\mathbb{Z}_n$ — это взаимно простые с n.
- Их всего $\phi(n)$ — функция Эйлера. Если p и q простые, то

$$\phi(p) = p - 1, \qquad \phi(pq) = (p-1)(q-1).$$

Арифметика по модулю n

- $\mathbb{Z}_n^+$ — это группа по сложению.
- $\mathbb{Z}_n^*$ — это группа по умножению.
- Сколько элементов в $\mathbb{Z}_n^*$?
- Если p — простое, то $\mathbb{Z}_p$ — это поле: у каждого элемента, кроме нуля, есть обратный по умножению.
- Над полем верны полезные факты из алгебры: например, над полем многочлен степени d имеет не более d корней.

Арифметика по модулю n

- На всякий случай ещё вспомним, что бывают конечные поля с p^m элементами.
- Их можно рассматривать как поля многочленов по модулю того или иного неприводимого многочлена.
- Например, поле $\mathbb{F}_{16}$ состоит из следующих элементов:

$$
\begin{array}{llll}
0, & x^2 & x^3 & x^2+x^3 \\
1 & x^2+1 & x^3+1 & x^2+x^3+1 \\
x & x^2+x & x^3+x & x^3+x^2+x \\
x+1 & x^2+x+1 & x^3+x+1 & x^3+x^2+x+1
\end{array}
$$

- Операции производятся по модулю x^4+x+1 (или x^4+x^3+1, или $x^4+x^3+x^2+1$ — получится одно и то же поле).

Малая теорема Ферма

- Если p простое, то для любого a $a^p \equiv a \pmod{p}$, а для любого a, взаимно простого с p, $a^{p-1} \equiv 1 \pmod{p}$.
- Соответственно, для простого p и любых m и n

$$\text{если } m \equiv n \pmod{p-1}, \quad \text{то } \forall a\ a^m \equiv a^n \pmod{p}.$$

- Теорема Эйлера — для любого n и любого a, взаимно простого с n,

$$a^{\Phi(n)} \equiv 1 \pmod{n}.$$

Алгоритм Евклида

- Алгоритм Евклида: классический — вычисляет gcd.
- Кроме $d = \gcd(a, b)$, вычисляет ещё два числа x и y, такие, что $ax + by = d$.
- Как применить алгоритм Евклида, чтобы найти $a^{-1} \pmod{n}$?
- Найти такие x и y, что $ax + ny = d$, где $d = \gcd(a, n)$.
- Если $d > 1$, то a необратимо в $\mathbb{Z}_p$; если $d = 1$, то $x = a^{-1} \pmod{n}$.

Возведение в степень

- Если есть два числа a и b по модулю n, и мы хотим вычислить $a^b \pmod n$, то можно вычислить

$$a^2 \pmod n,\ a^3 \pmod n,\ \ldots$$

- Здесь $b-1$ умножение по модулю n.
- Можно ли лучше?

Двоичный алгоритм возведения в степень

- Можно сделать так: запишем b как строку битов. Потом будем возводить a в квадрат, домножая на a там, где у b биты равны 1. Например:

$$b = 9_{10} = 1001_2 \quad \Rightarrow \quad a^b = ((a^2)^2)^2 \cdot a, \text{ 4 умножения.}$$

$$b = 65537_{10} = 10000000000000001_2 \quad \Rightarrow$$

$$\Rightarrow \quad a^b = (((a^2)^2)\ldots)^2 \cdot a, \text{ 17 умножений.}$$

- 17 значительно меньше, чем 65536.

Квадратные корни

- Теперь давайте наоборот. Как по $x^2 \pmod{p}$ найти $x \pmod{p}$?
- Во-первых, не всякое число является квадратом по модулю p. Те, которые являются, называются *квадратичными вычетами*.
- В $\mathbb{Z}_p^*$ вычетов столько же, сколько невычетов, а именно $\frac{p-1}{2}$. Почему?

Квадратные корни

- Рассмотрим $1^2, 2^2, \ldots, \left(\frac{p-1}{2}\right)^2$.
- Поскольку $n^2 \equiv (p-n)^2 \pmod{n}$, всего вычетов не больше $\frac{p-1}{2}$.
- Пусть их меньше. Тогда для некоторых $1 \leq i, j \leq \frac{p-1}{2}$

$$i^2 \equiv (-i)^2 \equiv j^2 \equiv (-j)^2 \pmod{p}.$$

- Иначе говоря, у уравнения $x^2 \equiv i^2 \pmod{p}$ четыре разных корня.
- Но $\mathbb{Z}_p$ — поле, и у него не может быть больше двух корней.

Символ Лежандра

- Символ Лежандра:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & a \equiv 0 \pmod p, \\ 1, & a \not\equiv 0 \pmod p, \text{ и для некоторого } x\ x^2 \equiv 0, \\ -1, & a \not\equiv 0 \pmod p, \text{ и такого } x \text{ не существует.} \end{cases}$$

- Для простого p

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p.$$

Символ Лежандра

- Кроме того,

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right),$$

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}\left(\frac{q}{p}\right).$$

- Это позволяет построить алгоритм для вычисления символа Лежандра $\left(\frac{a}{p}\right)$:
  - разложить $\left(\frac{a}{p}\right)$ в произведение $\left(\frac{p_1}{p}\right)\dots\left(\frac{p_m}{p}\right)$;
  - заменить на $p_i \pmod{p}$, перевернуть, повторить.

Квадратный корень

- Теперь возвращаемся к квадратному корню. Пусть дано простое p и $a \in \mathbb{Z}_p$.
- Если $p \equiv 3 \pmod 4$, то корень ищется как

$$x \equiv a^{(p+1)/4} \pmod n.$$

- Действительно,

$$1 = \left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p.$$

- Значит,

$$x^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \equiv a \pmod p.$$

Квадратный корень

- Для $p \equiv 1 \pmod 4$ — вероятностный алгоритм.
- Рассмотрим многочлен $x^{(p-1)/2} - 1$. Он степени $\frac{p-1}{2}$, его корни — все квадратичные вычеты по модулю p, и только они.
- Теперь рассмотрим многочлен $f(x) \equiv x^2 - a \equiv (x - r)(x + r) \pmod p$. Подставим

$$f(x - \delta) \equiv (x - (\delta - r))(x - (\delta + r)) \pmod p.$$

- Факт (без доказательства): для половины δ одно из значений $(\delta - r)$, $(\delta + r)$ является вычетом, а другое — нет.
- Выберем δ случайно и подсчитаем $\gcd(f(x - \delta), x^{(p-1)/2} - 1)$ (как многочленов).
- Тогда с вероятностью $1/2$ мы получим корень из a.

Для составных n

- Пусть, например, $n = pq$. Алгоритм вычисления квадратного корня из a по модулю n.
  1. Найти корни $(r, -r)$ числа a по модулю p.
  2. Найти корни $(s, -s)$ числа a по модулю q.
  3. Найти алгоритмом Евклида такие c и d, что $cp + dq = 1$.
  4. Вычислить $x = rdq + scp \pmod{n}$ и $y = rdq - scp \pmod{n}$.
  5. Вернуть $(\pm x, \pm y)$.
- Иначе говоря, мы можем вычислять квадратные корни, если умеем раскладывать n на множители.

Следствие

- Вычисление квадратного корня потребовало уметь раскладывать a на множители.
- Без этого даже не проверить, является ли a вычетом.
- А можно ли наоборот? Можно ли разложить число на множители, умея вычислять квадратные корни по его модулю?

Разложение на множители через корни

- Можно! Предположим, что мы умеем выдавать некий квадратный корень по модулю n.
- Возьмём случайное x, вычислим $a = x^2$ и подадим алгоритму.
- Если мы получили $\pm x$, повторим операцию. А если получили $y \neq \pm x$, то получилось, что

$$x^2 \equiv y^2 \pmod{n}, \text{ но } y \neq \pm x \pmod{n}.$$

- Это значит, что n делит $x^2 - y^2 = (x - y)(x + y)$, но при этом не делит либо $x - y$, либо $x + y$.
- Значит, $\gcd(x - y, n)$ — нетривиальный делитель n.

Постановка задачи

- Теперь поставим более сложную задачу — найти логарифм.
- *Дискретный логарифм*: по простому числу p, числу $a \in \mathbb{Z}_p^*$, порождающему $\mathbb{Z}_p^*$, и числу $b \in \mathbb{Z}_p^*$ найти такое $0 \leq x \leq p-2$, что

$$a^x \equiv b \pmod{p}.$$

- *Обобщённый дискретный логарифм*: то же в произвольной циклической группе G: по генератору $a \in G$ и $b \in G$ найти такой x, что $a^x = b$.

Замечания

- Сложность не зависит от генератора a; для другого генератора a'

$$a^x = b = a'^y = (a^z)^y, \text{ и } \log_{a'} b = \log_a b(\log_a a')^{-1}.$$

- Но сложность зависит от представления группы, т.е. для изоморфных групп сложность дискретного логарифма может быть разной. Почему?
- Потому что любая циклическая группа изоморфна $\mathbb{Z}_n^+$ для некоторого n.
- Дискретный логарифм в $\mathbb{Z}_n^+$ — это значит найти такой x, что $ax = b \pmod{n}$. Наверное, это не так уж сложно...

Замечания

- Алгоритмы для задачи дискретного логарифма делятся на три группы:
  1. Работающие для любых групп.
  2. Работающие для любых групп, но эффективные для «гладких» (когда порядок группы имеет маленькие простые делители).
  3. Эффективные только для некоторых групп.

Итоги

- Мы теперь умеем в $\mathbb{Z}_n$:
    - быстро возводить в степень;
    - находить a^{-1};
    - использовать алгоритм Евклида;
    - применять равенство $a^{\phi(n)} \equiv 1 \pmod{n}$.
- Мы выяснили, что умеем раскладывать n на множители тогда и только тогда, когда умеем вычислять по модулю n квадратные корни. $O(\sqrt[3]{n})$
- И узнали о задаче дискретного логарифма. $O(\sqrt[3]{n})$

Построение публичных криптосистем

- Основная цель: Построение CCA стойкую систему шифрования на публичных ключах.
- Метод: Использование однонаправленных функций с секретной дверью (Trapdoor functions (TDF))

Функции с секретной дверью

- Определение: Функция с секретной дверью X->Y тройка алгоритмов (G, F, F^{-1}), таких что:

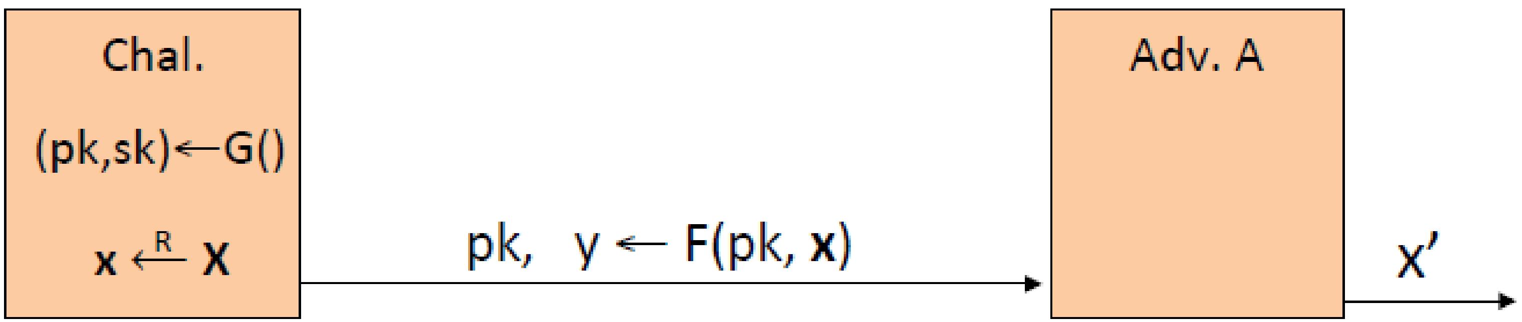
  - G(): рандомизированный алгоритм генерации ключей pk, sk.
  - F(pk,·): детерминированный алгоритм , который определяет отображение X->Y
  - F^{-1}(sk,·): определяет функцию Y->X, которая обращает F(pk,·)

- Более точно:
  - Для $\forall(pk, sk)$ полученных от G

$$\forall x \in X \; F^{-1}\big(sk, F(pk, x)\big) = x$$

Стойкие функции с секретными дверями

- Набор алгоритмов (G, F, F^{-1}) является стойким, если *F(pk,x)* – однонаправленная функция:
  - Может быть вычислена, но не может быть найдена обратная



- Определение: (G, F, F^{-1}) является стойкой TDF, если для всех алгоритмов A верно

$$Adv_{OW}[A, F] = \Pr\{x = x'\} < neg$$

Публичные криптосистемы на основе TDF

- (G, F, F^{-1}): стойкая TDF $X \to Y$
- (E_s, D_s) система симметричного аутентифицированного шифрования над (K, M, C)
- H: $X \to K$ – функция хеширования.

- Будем строить публичную криптосистему (G, E, D)

  - Генерация ключей из TDF

Публичные криптосистемы на основе TDF

- (G, F, F^{-1}): стойкая TDF $X \rightarrow Y$
- (E_S, D_S) система симметричного аутентифицированного шифрования над (K, M, C)
- $H: X \rightarrow K$ – функция хеширования.

E(pk, m) :

$$x \xleftarrow{R} X, \qquad y \leftarrow F(pk, x)$$

$$k \leftarrow H(x), \qquad c \leftarrow E_s(k, m)$$

$$\text{output} \quad (y, c)$$

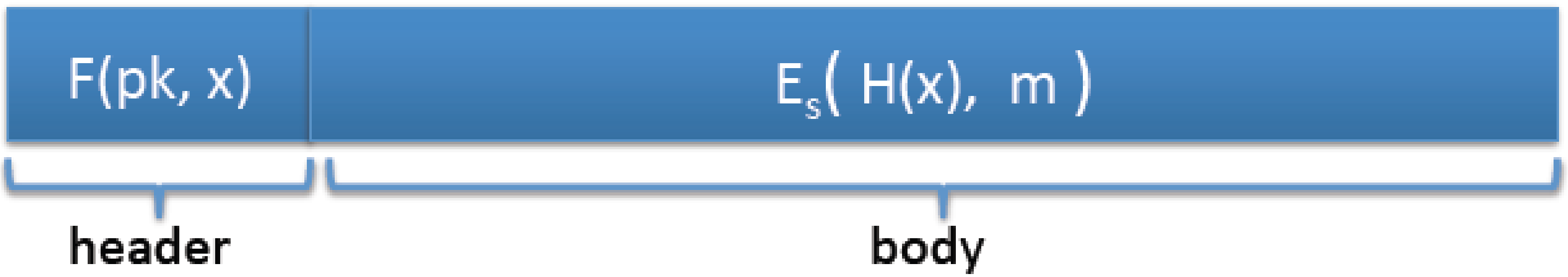
D(sk, (y,c)) :

$$x \leftarrow F^{-1}(sk, y),$$

$$k \leftarrow H(x), \qquad m \leftarrow D_s(k, c)$$

$$\text{output} \quad m$$

- Схема шифрования:



- Доказательство безопасности:
- Если **(G, F, F^{-1})**: стойкая TDF, **(E_s,D_s)** обеспечивает аут. шифрование и **H**: X→K – «случайный оракул», тогда **(G, E, D)** будет CCA^{ro} стойкий

Некорректное использование TDF

- **Нельзя** использовать шифрование непосредственно с открытым текстом

<u>E(pk, m)</u> :

output $c \longleftarrow F(pk, m)$

<u>D(sk, c)</u> :

output $F^{-1}(sk, c)$

- **Проблемы:**
  - Детерминированный алгоритм не может быть CC
  - Существует множество атак

RSA перестановка с секретной дверью

- Пусть N=pq

  - Z_N={0,1,2, … N-1}; $(Z_N)^*$-- множество обратимых элементов

- Факты: $x \in Z_N$ обратимы ←→ gsd(x,N) = 1
  - Количество элементов в $(Z_N)^*$ определяется

$$\varphi(N) = (p-1)(q-1) = N - p - q + 1$$

- Теорема Эйлера: $\boxed{\forall\, x \in (Z_N)^* \;:\; x^{\varphi(N)} = 1}$

Описание схемы

- **G()**: выбрать p,q по 1024 бит. Вычислить N=pq
- Выбрать значения d и e, такие что ed=1 mod $\varphi(N)$
- Возвращает pk=(N, e), sk = (d,N)

F(pk, x): $\mathbb{Z}_N^* \to \mathbb{Z}_N^*$; **RSA(x) = x^e** (in Z_N)

F^{-1}(sk, y) = y^d ; y^d = **RSA(x)^d** = x^{ed} = $x^{k\varphi(N)+1}$ = $\left(x^{\varphi(N)}\right)^k \cdot$ **x** = x

RSA допущение

For all efficient algs. A:

$$\Pr\left[\; A(N,e,y) = y^{1/e} \;\right] < \text{negligible}$$

where $\quad p,q \xleftarrow{R} \text{n-bit primes}, \quad N \leftarrow pq, \quad y \xleftarrow{R} Z_N^{\,*}$

Применение RSA

- Применение RSA в режиме кодовой книги:

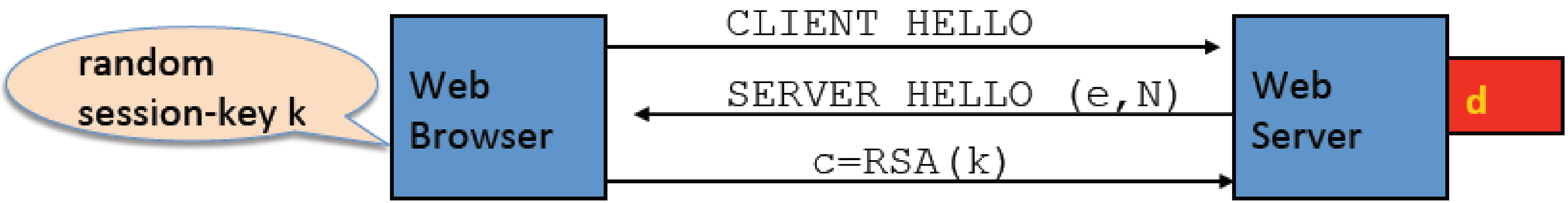
– public key: **(N,e)** Encrypt: $\mathbf{c} \longleftarrow \mathbf{m}^{\mathbf{e}}$ (in Z_N)

– secret key: **(N,d)** Decrypt: $\mathbf{c}^{\mathbf{d}} \longrightarrow \mathbf{m}$

- Нестойкая криптосистема!!
  - Не CC-стойкая + гомоморфные атаки.

- RSA не является системой шифрования.

Описание простой атаки на RSA



Если k 64 бита, то с вероятностью 20% его можно разложить на множители

$\mathbf{k = k_1 \cdot k_2}$ where $k_1, k_2 < 2^{34}$ (prob. ≈20%) then $\mathbf{c/k_1^e = k_2^e}$ in Z_N

Step 1: build table: $c/1^e, c/2^e, c/3^e, \ldots, c/2^{34e}$. time: 2^{34}

Step 2: for $k_2 = 0,\ldots, 2^{34}$ test if k_2^e is in table. time: 2^{34}