Криптография с открытым ключом

- Основные требования к алгоритмам асимметричного шифрования
- Криптоанализ алгоритмов с открытым ключом
- Основные способы использования алгоритмов с открытым ключом
- Требования к цифровой подписи
- Алгоритм RSA
- Алгоритм обмена ключа Диффи-Хеллмана
- Стандарт цифровой подписи DSS
- Стандарт цифровой подписи ГОСТ 3410

Основные требования к алгоритмам асимметричного шифрования

- Алгоритмы асимметричного шифрования, называемые также алгоритмами с открытым ключом, принципиально отличаются от алгоритмов симметричного шифрования. Шифрование с открытым ключом является **асимметричным**, поскольку **использует два различных ключа для шифрования и расшифрования**, в отличие от симметричного шифрования, в котором для шифрования и расшифрования используется один и тот же ключ. Алгоритмы с открытым ключом гораздо больше **основаны на свойствах математических функций**, чем алгоритмы симметричного шифрования, использующие в основном только операции подстановки и перемещения. Наличие двух ключей имеет важное применение в таких областях, как **аутентификация, распределение ключа и конфиденциальность**.

Основные требования к алгоритмам асимметричного шифрования

- Алгоритмы с открытым ключом разрабатывались для того, чтобы решить две наиболее трудные задачи, возникшие при использовании симметричного шифрования.
- **Первой задачей является распределение ключа.** При симметричном шифровании требуется, чтобы обе стороны уже имели общий ключ, который каким-то образом должен быть им заранее передан. Диффи, один из основоположников шифрования с открытым ключом, заметил, что это требование отрицает всю суть криптографии, основное назначение которой поддерживать секретность коммуникаций.
- **Второй задачей является необходимость** создания таких механизмов, при использовании которых невозможно было бы подменить кого-либо из участников, т.е. нужен **аналог подписи**, которая используется в реальном мире. Такой аналог обычно называется цифровой или электронной подписью (англ. вариант – **Digital Signature**). При использовании коммуникаций для решения широкого круга задач, например в коммерческих и частных целях, электронные сообщения и документы должны иметь эквивалент подписи, содержащейся в бумажных документах. Необходимо создать метод, при использовании которого все участники будут убеждены, что электронное сообщение было послано конкретным участником. Это более сильное требование, чем аутентификация с использованием пароля или общего секрета.

Основные требования к алгоритмам асимметричного шифрования

- Сначала рассмотрим общие черты алгоритмов шифрования с открытым ключом и требования к этим алгоритмам. Определим требования, которым должен соответствовать алгоритм, использующий один ключ для шифрования, другой ключ - для расшифрования, и при этом вычислительно невозможно определить ключ расшифрования, зная только алгоритм и ключ шифрования.
- Кроме того, некоторые алгоритмы, например RSA, имеют следующее свойство: каждый из двух ключей может использоваться как для шифрования, так и для расшифрования.
- Сначала рассмотрим алгоритмы, обладающие обеими характеристиками, а затем перейдем к алгоритмам открытого ключа, которые не обладают вторым свойством.
- При описании симметричного шифрования и шифрования с открытым ключом будем использовать следующую терминологию. Ключ, используемый в симметричном шифровании, будем называть **секретным ключом**. Два ключа, используемые при шифровании с открытым ключом, будем называть **открытым ключом (Key Public – KU)** и **закрытым ключом (Key Private – KR)**. Закрытый ключ держится в секрете, но называть его будем закрытым ключом, а не секретным, чтобы избежать путаницы с ключом, используемым в симметричном шифровании. Закрытый ключ будем обозначать **KR**, открытый ключ – **KU**.

Основные требования к алгоритмам асимметричного шифрования

- Будем предполагать, что все участники имеют доступ к открытым ключам друг друга, а закрытые ключи создаются локально каждым участником и, следовательно, распределяться не должны.
- В любое время участник может изменить свой закрытый ключ и опубликовать составляющий пару открытый ключ, заменив им старый открытый ключ.
- Диффи и Хеллман описывают требования, которым должен удовлетворять алгоритм шифрования с открытым ключом.

1. Вычислительно легко создавать пару (открытый ключ **KU**, закрытый ключ **KR**).
2. Вычислительно легко, имея открытый ключ и незашифрованное сообщение **M**, создать соответствующее зашифрованное сообщение:

$$C = E_{KU}[M]$$

Основные требования к алгоритмам асимметричного шифрования

3. Вычислительно легко расшифровать сообщение, используя закрытый ключ:

$$M = D_{KR}[C] = D_{KR}[E_{KU}[M]]$$

4. Вычислительно невозможно, зная открытый ключ **KU**, определить закрытый ключ **KR**.
5. Вычислительно невозможно, зная открытый ключ **KU** и зашифрованное сообщение **C**, восстановить исходное сообщение **M**.

Основные требования к алгоритмам асимметричного шифрования

Можно добавить шестое требование, хотя оно не выполняется для всех алгоритмов с открытым ключом:

6. Шифрующая и расшифровывающая функции могут применяться в любом порядке:

$$M = E_{KU}[D_{KR}[M]]$$

Основные требования к алгоритмам асимметричного шифрования

- Односторонней функцией называется такая функция, у которой каждый аргумент имеет единственное обратное значение, при этом вычислить саму функцию легко, а вычислить обратную функцию трудно.

  $Y = f(X)$ – вычислительно легко

  $X = f^{-1}(Y)$ – вычислительно трудно

- В данном случае термин «вычислительно легко» означает полиномиальную сложность от длины входа. Таким образом, если алгоритму на вход подается значение длиной n битов, то время вычисления функции пропорционально n^a, где a – фиксированная константа. Таким образом, говорят, что алгоритм принадлежит классу полиномиальных алгоритмов **P**. Термин «вычислительно трудно» означает более сложное понятие. В общем случае будем считать, что проблему решить невозможно, если усилия для ее решения больше полиномиального времени от величины входа. Например, если длина входа n битов, и время вычисления функции пропорционально 2^n, то это считается вычислительно невозможной задачей. К сожалению, тяжело определить, проявляет ли конкретный алгоритм такую сложность. Более того, традиционные представления о вычислительной сложности фокусируются на поведении алгоритмов в худшем случае или в среднем случае. Это неприемлемо для криптографии, где требуется невозможность инвертировать функцию для всех или почти всех значений входов.

Основные требования к алгоритмам асимметричного шифрования

Таким образом, односторонняя функция с люком принадлежит семейству односторонних функций f_k таких, что

- $Y = f_k(X)$ - легко, если k и X известны
- $X = f_k^{-1}(Y)$ - легко, если k и Y известны
- $X = f_k^{-1}(Y)$ - трудно, если Y известно, но k неизвестно

Криптоанализ алгоритмов с открытым ключом

- Как и в случае симметричного шифрования, алгоритм шифрования с открытым ключом уязвим для **лобовой атаки**. Контрмера стандартная: использовать ключи большей длины.
- Криптосистема с открытым ключом **использует не инвертируемые математические функции**. Сложность вычислений таких функций не является линейной от количества битов ключа, а возрастает быстрее, чем ключ. Таким образом, размер ключа должен быть достаточно большим, чтобы сделать лобовую атаку непрактичной, и достаточно маленьким для возможности практического шифрования. На практике размер ключа делают таким, чтобы **лобовая атака была непрактичной**, но в результате скорость шифрования оказывается достаточно медленной для использования алгоритма в общих целях. Поэтому шифрование с открытым ключом в настоящее время в основном ограничивается приложениями управления ключом и создания подписи, в которых требуется шифрование или подписывание небольшого блока данных.

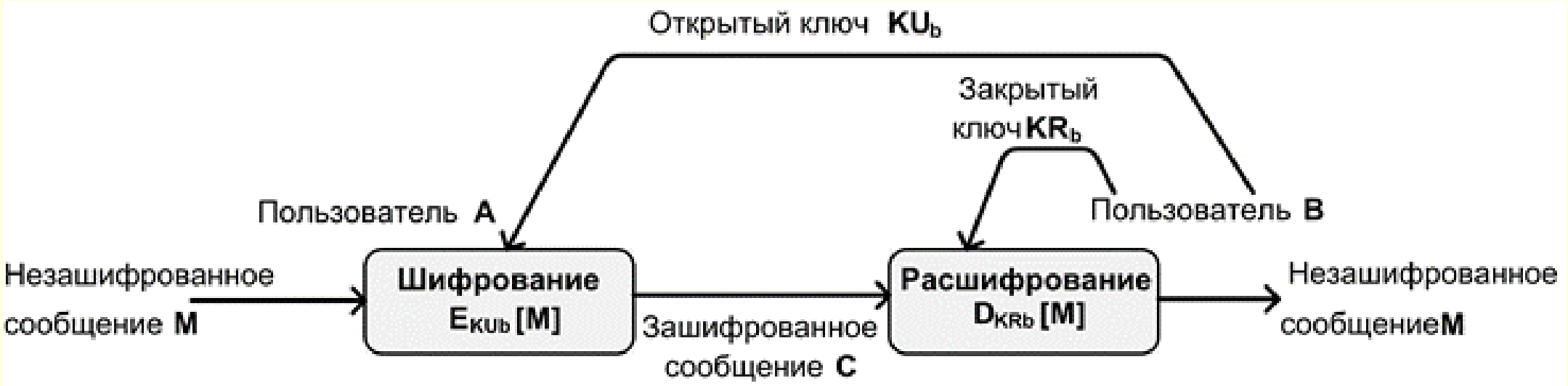
Криптоанализ алгоритмов с открытым ключом

- Другая форма атаки состоит в том, чтобы **найти способ вычисления закрытого ключа, зная открытый ключ**. Невозможно математически доказать, что данная форма атаки исключена для конкретного алгоритма открытого ключа. Таким образом, любой алгоритм, включая широко используемый алгоритм RSA, является подозрительным.
- Наконец, существует форма атаки, специфичная для способов использования систем с открытым ключом. Это **атака вероятного сообщения**. Предположим, например, что посылаемое сообщение состоит исключительно из 56-битного ключа сессии для алгоритма симметричного шифрования. Противник может зашифровать все возможные ключи, используя открытый ключ получателя. В этом случае атака сводится к лобовой атаке на 56-битный симметричный ключ. Защита от подобной атаки состоит в добавлении определенного количества случайных битов в простые сообщения.

Основные способы использования алгоритмов с открытым ключом

Основными способами использования алгоритмов с открытым ключом являются шифрование / расшифрование, создание и проверка подписи и обмен ключа.

- **Шифрование** с открытым ключом состоит из следующих шагов:

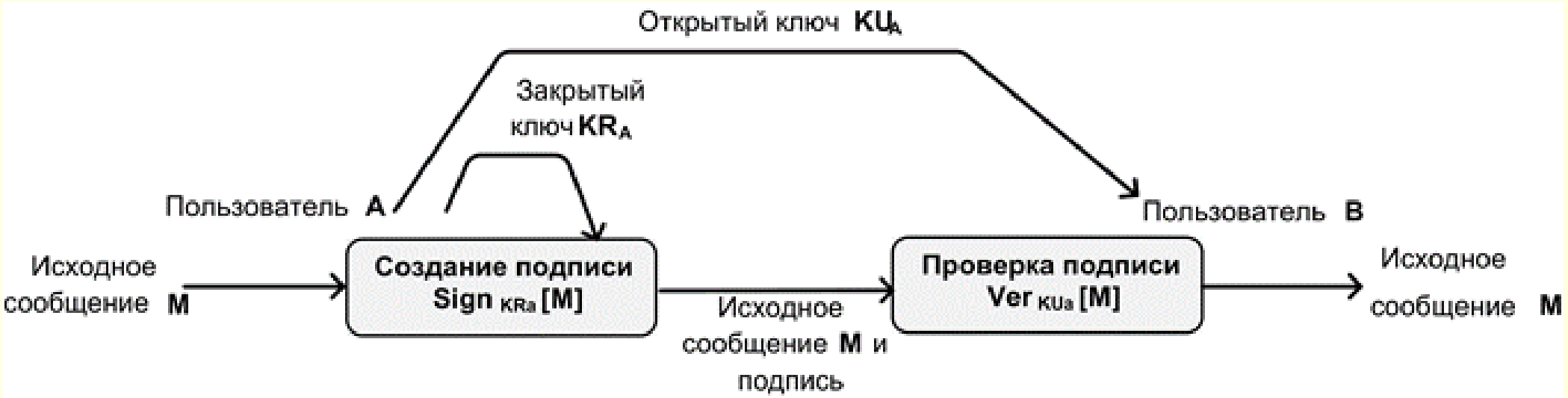


Основные способы использования алгоритмов с открытым ключом

1. Пользователь **B** создает пару ключей $\mathbf{KU_b}$ и $\mathbf{KR_b}$, которые могут использоваться для шифрования и расшифрования передаваемых сообщений.
2. Пользователь **B** передает пользователю **A** некоторым надежным способом свой ключ шифрования, т.е. открытый ключ $\mathbf{KU_b}$. Составляющий пару закрытый ключ $\mathbf{KR_b}$ держится в секрете.
3. Если **A** хочет послать конфиденциальное сообщение **B**, он шифрует сообщение, используя открытый ключ **B** $\mathbf{KU_b}$.
4. Когда **B** получает сообщение, он расшифровывает его, используя свой закрытый ключ $\mathbf{KR_b}$. Никто другой не сможет расшифровать сообщение, так как этот закрытый ключ знает только **B**.
5. Если пользователь **B** надежно хранит свой закрытый ключ, никто не сможет расшифровать передаваемые сообщения.

Основные способы использования алгоритмов с открытым ключом

- **Создание и проверка подписи** состоит из следующих шагов:



Основные способы использования алгоритмов с открытым ключом

1. Пользователь **A** создает пару ключей KR_A и KU_A, которые могут использоваться для создания и проверки подписи.
2. Пользователь **A** передает пользователю **B** некоторым надежным способом свой ключ проверки подписи, т.е. открытый ключ KU_A. Составляющий пару закрытый ключ KR_A держится в секрете.
3. Если **A** хочет послать подписанное сообщение пользователю **B**, он создает подпись $Sign_{KRa}[M]$ этого сообщения, используя свой закрытый ключ KR_A.
4. Когда **B** получает подписанное сообщение, он проверяет подпись $Ver_{KUa}[M]$, используя открытый ключ KU_A пользователя **A**. Никто другой не может подписать сообщение, так как этот закрытый ключ знает только **A**.
5. До тех пор, пока пользователь **A** надежно хранит свой закрытый ключ, его подписи достоверны. Кроме того, невозможно изменить сообщение, не имея доступа к закрытому ключу **A**; тем самым обеспечивается аутентификация отправителя и целостность передаваемых данных. Все алгоритмы создания и проверки подписи имеют большую вычислительную нагрузку, так как связаны с возведением в большие степени. Более эффективным способом является подписывание небольшого блока битов, который является функцией от сообщения. Такой блок, называемый аутентификатором, должен обладать таким свойством, что любое изменение сообщения с большой вероятностью приводит к изменению его аутентификатора. Этот аутентификатор подписывается закрытым ключом отправителя, т.е. создается цифровая подпись. Далее эта технология будет рассматриваться в деталях.

Основные способы использования алгоритмов с открытым ключом

- Важно подчеркнуть, что описанный процесс создания подписи не обеспечивает конфиденциальность. Это означает, что сообщение, посланное таким способом, невозможно изменить, но можно подсмотреть. Это очевидно в том случае, если подпись основана на аутентификаторе, так как само сообщение передается в явном виде. Но даже если осуществляется подписывание всего сообщения, конфиденциальность не обеспечивается, так как любой может получить исходное сообщение, используя открытый ключ отправителя.
- Аутентификация защищает двух участников, которые обмениваются сообщениями, от взаимодействия с некоторой третьей стороной. Однако аутентификация, выполняемая с использованием общего секрета (пароля), не защищает участников друг от друга, тогда как и между ними тоже могут возникать определенные формы споров.
- Например, предположим, что **A** посылает **B** аутентифицированное сообщение, и аутентификация осуществляется на основе общего секрета. Рассмотрим возможные проблемы, которые могут при этом возникнуть:
- **B** может подделать сообщение и утверждать, что оно пришло от **A. B** достаточно просто создать сообщение и присоединить аутентификационный код, используя ключ, который разделяют **A** и **B**.
- **A** может отрицать, что он посылал сообщение **B**. Так как **B** может подделать сообщение, у него нет способа доказать, что **A** действительно посылал его.

Основные способы использования алгоритмов с открытым ключом

- В ситуации, когда обе стороны не доверяют друг другу, необходимо нечто большее, чем аутентификация на основе общего секрета. Возможным решением подобной проблемы является использование цифровой подписи. Цифровая подпись должна обладать следующими свойствами:

1. Должна быть возможность проверить автора, дату и время создания подписи.
2. Должна быть возможность аутентифицировать содержимое во время создания подписи.
3. Подпись должна быть проверяема третьей стороной для разрешения споров.

- Таким образом, создание цифровой подписи включает сервис аутентификации отправителя.

Основные способы использования алгоритмов с открытым ключом

- **Обмен ключей:** две стороны взаимодействуют для обмена ключом сессии, который в дальнейшем можно использовать в алгоритме симметричного шифрования.
- Некоторые алгоритмы можно задействовать тремя способами, в то время как другие могут использоваться одним или двумя способами.
- Перечислим наиболее популярные алгоритмы с открытым ключом и возможные способы их применения.

Алгоритм	Шифрование/ расшифрование	Цифровая подпись	Обмен ключей
RSA	Да; непригоден для больших блоков	Да	Да
DSS	Нет	Да	Нет
Диффи-Хеллман	Нет	Нет	Да

Требования к цифровой подписи

На основании этих свойств можно сформулировать следующие требования к цифровой подписи:

1. Подпись должна быть битовым образцом, который зависит от подписываемого сообщения.
2. Подпись должна использовать некоторую уникальную информацию отправителя для предотвращения подделки или отказа.
3. Создавать цифровую подпись должно быть относительно легко.
4. Должно быть относительно легко проверять цифровую подпись.
5. Должно быть вычислительно невозможно подделать цифровую подпись как созданием нового сообщения для существующей цифровой подписи, так и созданием ложной цифровой подписи для данного сообщения.
6. Цифровая подпись должна быть достаточно компактной и не занимать много памяти.

Сильная хеш-функция, подписанная закрытым ключом отправителя, удовлетворяет перечисленным требованиям.

Алгоритм RSA

- Одним из первых результатов был алгоритм, разработанный в 1977 году Роном Ривестом, Ади Шамиром и Леном Адлеманом и опубликованный в 1978 году. С тех пор алгоритм Rivest-Shamir-Adleman (RSA) широко применяется практически во всех приложениях, использующих криптографию с открытым ключом.
- Алгоритм основан на использовании того факта, что задача факторизации является трудной, т.е. легко перемножить два числа, в то время как не существует полиномиального алгоритма нахождения простых сомножителей большого числа.
- Алгоритм RSA представляет собой блочный алгоритм шифрования, где зашифрованные и незашифрованные данные являются целыми между **`0`** и **`n-1`** для некоторого **`n`**.

Алгоритм RSA

- Алгоритм, разработанный Ривестом, Шамиром и Адлеманом, использует выражения с экспонентами. Данные шифруются блоками, каждый блок рассматривается как число, меньшее некоторого числа **n**. В результате шифрования числа **M**, **M<n**, получается число **C**.

$$C = M^e \pmod n$$

- Расшифрование выполняется следующим образом:

$$M = C^d \pmod n = (M^e)^d \pmod n = M^{ed} \pmod n$$

- Как отправитель, так и получатель должны знать число **n**. Отправитель знает число, получатель знает число **d**. Таким образом, открытый ключ есть **KU={e, n}** и закрытый ключ есть **KR={d, n}.**

Алгоритм RSA

При этом должны выполняться следующие условия:

1. Возможность найти значения **e**, **d** и **n** такие, что $M^{ed} = M \pmod{n}$ для всех $M < n$.
2. Относительно легко вычислить M^e и C^d для всех значений $M < n$.
3. Невозможность определить **d**, зная **e** и **n**.

- Сначала рассмотрим первое условие. Нам необходимо выполнение равенства:

$$M^{ed} = M \pmod{n}$$

Алгоритм RSA

- Рассмотрим некоторые математические понятия, свойства и теоремы, которые позволят нам определить **e, d** и **n**.

1. Если $a \cdot b = a \cdot c \pmod{n}$, то $b = c \pmod{n}$, если **a** и **n** взаимнопростые, т.е **НОД(a,n) = 1**.
2. Обозначим Z_p - все числа, взаимно простые с **p** и меньшие **p**. Если **p** - простое, то Z_p - это все остатки. Обозначим w^{-1} такое число, что

$$w \cdot w^{-1} = 1 \pmod{p}.$$

Тогда $\forall w \in Z_p \ \exists z: \ w \cdot z = 1 \pmod{p}$

- Доказательство этого следует из того, что т.к. **w** и **p** взаимно простые, то при умножении всех элементов Z_p на **w** по модулю **p** остатками будут все элементы Z_p, возможно, переставленные. Таким образом, хотя бы один остаток будет равен **1**.

Алгоритм RSA

1. Определим **функцию Эйлера** следующим образом: $\phi(n)$ - число положительных чисел, меньших n и взаимнопростых с n. Если p - простое, то $\phi(p) = p-1$.
   - Покажем, что если p и q - простые, то $\phi(p \cdot q) = (p-1) \cdot (q-1)$.
   - Перечислим числа, меньшие $p \cdot q$, которые не являются взаимнопростыми с $p \cdot q$:
   - Числа, которые делятся на p: $\{p, 2 \cdot p, \ldots, (q-1) \cdot p\}$. Таких чисел $(q - 1)$.
   - Числа, которые делятся на q: $\{q, 2 \cdot q, \ldots, (p-1) \cdot q\}$. Таких чисел $(p - 1)$.

Таким образом

$$\phi(p \cdot q) = p \cdot q - 1 - [(q-1)+(p-1)] = p \cdot q - (p+q) + 1 = (p-1) \cdot (q-1)$$

Алгоритм RSA

Теорема Ферма

$a^{n-1} = 1 \pmod n$, если n - простое.

Если все элементы Z_n умножить на a по модулю n, то в результате получим все элементы Z_n, быть может, в другом порядке. Рассмотрим следующие числа:

$\{a \pmod n, 2 \cdot a \pmod n, \ldots, (n-1) \cdot a \pmod n\}$ являются числами $\{1, 2, \ldots, (n-1)\}$, быть может, в некотором другом порядке. Теперь перемножим по модулю n числа из этих двух множеств.

$$[(a \pmod n) \cdot (2a \pmod n) \cdot \ldots \cdot (n-1)a \pmod n] \pmod n = (n-1)! \pmod n$$

$$(n-1)!\ a^{n-1} = (n-1)! \pmod n$$

n и $(n-1)!$ являются взаимнопростыми, если n – простое.

Следовательно, $a^{n-1} = 1 \pmod n$.

Алгоритм RSA

Теорема Эйлера

$a^{\phi(n)} = 1 \pmod n$ для всех взаимнопростых a и n.

Это верно, если n - простое, т.к. в этом случае $\phi(n) = n-1$. Рассмотрим множество

$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}$.

Теперь умножим по модулю n каждый элемент этого множества на a. Получим множество

$S = \{a \cdot x_1 \pmod n, a \cdot x_2 \pmod n, \ldots, a \cdot x_{\phi(n)} \pmod n\}$.

Это множество является перестановкой множества R по следующим причинам.

Алгоритм RSA

Так как **a** является взаимнопростым с **n** и x_i являются взаимнопростыми с **n**, то $a \cdot x_i$ также являются взаимнопростыми с **n**. Таким образом, **S** - это множество целых, меньших **n** и взаимнопростых с **n**.

В **S** нет дублей, т.к. если $a \cdot x_i \pmod n = a \cdot x_j \pmod n \Rightarrow x_i = x_j$.

Следовательно, перемножив элементы множеств **S** и **R**, получим:

$$\prod_{i=1}^{\phi(n)} a \cdot x_i \pmod n = \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$\prod_{i=1}^{(n)} a \cdot x_i = \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$a^{\phi(n)} \cdot \prod_{i=1}^{\phi(n)} x_i = \prod x_i \pmod n$$

$$a^{\phi(n)} = 1 \pmod n$$

Алгоритм RSA

Теперь рассмотрим сам алгоритм RSA. Пусть p и q - простые.

$$n = p \cdot q$$

Надо доказать, что $\forall M < n: M^{\phi(n)} = M^{(p-1)\cdot(q-1)} = 1 \pmod{n}$

Если $НОД\ (M, n) = 1$, то равенство выполняется. Теперь предположим, что $НОД\ (M, n) \neq 1$, т.е. $НОД\ (M, p \cdot q) \neq 1$. Пусть $НОД\ (M, p) \neq 1$, т.е. $M = c \cdot p$, следовательно $НОД\ (M, q) = 1$, так как в противном случае $M = c \cdot p$ и $M = l \cdot q$, но по условию $M < p \cdot q$.

Следовательно,

$$M^{\phi(q)} = 1 \pmod{q}$$

$$(M^{\phi(q)})^{\phi(p)} = 1 \pmod{q}$$

$$M^{\phi(n)} = 1 \pmod{q}$$

По определению модуля это означает, что $M^{\phi(n)} = 1 + k \cdot q$. Умножим обе части равенства на $M = c \cdot p$. Получим $M^{\phi(n)+1} = c \cdot p + k \cdot q \cdot c \cdot p$.

$$M^{\phi(n)} = 1 \pmod{n}$$

Или

$$M^{\phi(n)+1} = M \pmod{n}$$

Алгоритм RSA

Таким образом, следует выбрать **e** и **d** такие, что

$$e \cdot d = 1 \pmod{\phi(n)}$$

Или

$$e = d^{-1} \pmod{\phi(n)}$$

e и **d** являются взаимнообратными по умножению по модулю **ϕ(n)**. Заметим, что в соответствии с правилами модульной арифметики, такие взаимнообратные элементы существуют только в том случае, если **d** (и, следовательно, **e**) являются взаимнопростыми с **ϕ(n)**. Таким образом, **НОД(ϕ(n), d) = 1**.

Алгоритм RSA

Теперь рассмотрим все элементы алгоритма RSA.

p, q - два простых целых числа- закрыты, выбираемы.

$n = p \cdot q$ - открыто, вычисляемо.

d, НОД $(\phi(n), d) = 1$; $1 < d < \phi(n)$ - закрыто, выбираемо.

$e = d^{-1} \pmod{\phi(n)}$ - открыто, вычисляемо.

Алгоритм RSA

Закрытый ключ состоит из **{d, n}**, открытый ключ состоит из **{e, n}**. Предположим, что пользователь **A** опубликовал свой открытый ключ, и что пользователь **B** хочет послать пользователю **A** сообщение **M**. Тогда **B** вычисляет $C = M^e \pmod n$ и передает **C**. При получении этого зашифрованного текста пользователь **A** расшифрует вычислением $M = C^d \pmod n$.

Алгоритм RSA

Суммируем алгоритм RSA:

Создание ключей

Выбрать простые p и q

Вычислить $n = p \cdot q$

Выбрать d: $НОД\ (\phi(n), d) = 1;\ 1 < d < \phi(n)$

Вычислить e: $e = d^{-1} \pmod{\phi(n)}$

Открытый ключ $KU = \{e, n\}$

Закрытый ключ $KR = \{d, n\}$

Шифрование

Незашифрованное сообщение: $M < n$

Зашифрованное сообщение: $C = M^e \pmod{n}$

Расшифрование

Зашифрованное сообщение: C

Незашифрованное сообщение: $M = C^d \pmod{n}$

Алгоритм RSA

Рассмотрим конкретный пример:

Выбрать два простых числа: $p = 7,\ q = 17.$

Вычислить $n = p \cdot q = 7 \cdot 17 = 119.$

Вычислить $\phi(n) = (p - 1) \cdot (q - 1) = 96.$

Выбрать e так, чтобы e было взаимнопростым с $\phi(n) = 96$ и меньше, чем $\phi(n)$: $e = 5$.

Определить d так, чтобы $d \cdot e = 1 \pmod{96}$ и $d < 96$.

$d = 77$, так как $77 \cdot 5 = 385 = 4 \cdot 96 + 1$.

Результирующие ключи открытый $KU=\{5,119\}$ и закрытый $KR=\{77,119\}$.

Алгоритм RSA

Например, требуется зашифровать сообщение $\mathbf{M = 19}$.

$\mathbf{19^5 = 66 \pmod{119}; C = 66}$.

Для расшифрования вычисляется $\mathbf{66^{77} \pmod{119} = 19}$.

Алгоритм RSA

Вычислительные аспекты

1. **Шифрование / расшифрование**

   - Как шифрование, так и расшифрование включают возведение целого числа в целую степень по модулю **n**. При этом промежуточные значения будут громадными. Для того чтобы частично этого избежать, используется следующее свойство модульной арифметики:

$$a \pmod n \cdot b \pmod n \pmod n = a \cdot b \pmod n$$

   - Другая оптимизация состоит в эффективном использовании показателя степени, так как в случае RSA показатели степени очень большие. Предположим, что необходимо вычислить x^{16}. Прямой подход требует 15 умножений. Однако можно добиться того же конечного результата с помощью только четырех умножений, если использовать квадрат каждого промежуточного результата: x^2, x^4, x^8, x^{16}.

Алгоритм RSA

1. **Создание ключей**

   - Создание ключей включает следующие задачи:

- Определить два простых числа **p** и **q**.
- Выбрать **e** и вычислить **d**.

  - Прежде всего, рассмотрим проблемы, связанные с выбором **p** и **q**. Так как значение **n = p·q** будет известно любому потенциальному противнику, для предотвращения раскрытия **p** и **q** эти простые числа должны быть выбраны из достаточно большого множества, т.е. **p** и **q** должны быть большими числами. С другой стороны, метод, используемый для поиска большого простого числа, должен быть достаточно эффективным.

  - Алгоритм, который используется для нахождения простых чисел, выбирает случайное нечетное число из требуемого диапазона и проверяет, является ли оно простым. Если число не является простым, то опять выбирается случайное число до тех пор, пока не будет найдено простое.

Алгоритм RSA

Были разработаны различные тесты для определения того, является ли число простым. Это тесты вероятностные, то есть тест показывает, что данное число *вероятно* является простым. Несмотря на это проверка числа на таких тестах делает вероятность того, что число простой, близкой к единице. Если **n** «проваливает» тест, то оно не является простым. Если **n** «пропускает» тест, то **n** может как быть, так и не быть простым. Если **n** пропускает много таких тестов, то можно с высокой степенью достоверности сказать, что **n** является простым. Это достаточно долгая процедура, но она выполняется относительно редко: только при создании новой пары **(KU, KR)**.

Алгоритм RSA

На сложность вычислений также влияет то, какое количество чисел будет отвергнуто перед тем, как будет найдено простое число. Результат из теории чисел, известный как теорема простого числа, говорит, что простых чисел, расположенных около **n** в среднем одно на каждые **ln(n)** чисел. Таким образом, в среднем требуется проверить последовательность из **ln(n)** целых, прежде чем будет найдено простое число. Так как все четные числа могут быть отвергнуты без проверки, то требуется выполнить приблизительно **ln(n)/2** проверок. Например, если простое число ищется в диапазоне величин 2^{200}, то необходимо выполнить около $\ln(2^{200}) / 2 = 70$ проверок.

Алгоритм RSA

Выбрав простые числа p и q, далее следует выбрать значение e так, чтобы $НОД(\phi(n), e) = 1$ и вычислить значение d, $d = e^{-1} \pmod{\phi(n)}$.

Существует единственный алгоритм, называемый расширенным алгоритмом Евклида, который за фиксированное время вычисляет наибольший общий делитель двух целых и, если этот общий делитель равен единице, определяет инверсное значение одного по модулю другого. Таким образом, процедура состоит в генерации серии случайных чисел и проверке каждого относительно $\phi(n)$ до тех пор, пока не будет найдено число, взаимно простое с $\phi(n)$.

Возникает вопрос, как много случайных чисел придется проверить до тех пор, пока не найдется нужное число, которое будет взаимно простым с $\phi(n)$.

Результаты показывают, что вероятность того, что два случайных числа являются взаимно простыми, равна 0.6.

Алгоритм RSA

- Обсуждение криптоанализа

  Можно определить четыре возможных подхода для криптоанализа алгоритма RSA:

1. Лобовая атака: перебрать все возможные закрытые ключи.
2. Разложить n на два простых сомножителя. Это даст возможность вычислить $\phi(n)=(p-1)\cdot(q-1)$ и $d=e^{-1} \pmod{\phi(n)}$.
3. Определить $\phi(n)$ непосредственно, без начального определения p и q. Это также даст возможность определить $d=e^{-1} \pmod{\phi(n)}$.
4. Определить d непосредственно, без начального определения $\phi(n)$.

Алгоритм RSA

- Защита от лобовой атаки для RSA и ему подобных алгоритмов состоит в использовании большой длины ключа. Таким образом, чем больше битов в **e** и **d**, тем лучше. Однако, так как вычисления, связанные с возведением в степень, необходимы как при создании ключей, так и при шифровании / расшифровании, чем больше размер ключа, тем медленнее работает система.
- Большинство дискуссий о криптоанализе RSA фокусируется на задаче разложения **n** на два простых сомножителя. В настоящее время неизвестны алгоритмы, с помощью которых можно было бы разложить число на два простых множителя для очень больших чисел (т.е. несколько сотен десятичных цифр). Лучший из известных алгоритмов дает результат, пропорциональный:

$$L(n) = e^{\sqrt{\ln n \cdot \ln(\ln n)}}$$

Алгоритм RSA

- Пока не разработаны лучшие алгоритмы разложения числа на простые множители, можно считать, что величина **n** от 100 до 200 цифр в настоящее время является достаточно безопасной. На современном этапе считается, что число из 100 цифр может быть разложено на множители за время порядка двух недель. Для дорогих конфигураций (т.е. порядка $10 млн) число из 150 цифр может быть разложено приблизительно за год. Разложение числа из 200 цифр находится за пределами вычислительных возможностей. Например, даже если вычислительный уровень в 10^{12} операций в секунду достижим, что выше возможностей современных технологий, то потребуется свыше 10 лет для разложения на множители числа из 200 цифр с использованием существующих алгоритмов.
- Для известных в настоящее время алгоритмов задача определения **ϕ(n)** по данным **e** и **n**, по крайней мере сопоставима по времени с задачей разложения числа на множители.

Алгоритм RSA

Для того чтобы избежать выбора значения **n**, которое могло бы легко раскладываться на сомножители, на **p** и **q** должно быть наложено много дополнительных ограничений:

- **p** и **q** должны друг от друга отличаться по длине только несколькими цифрами. Таким образом, оба значения **p** и **q** должны быть от 1075 до 10100.
- Оба числа **(p – 1)** и **(q – 1)** должны содержать большие простые сомножители.
- **НОД(p–1, q–1)** должен быть маленьким.

Алгоритм Диффи-Хеллмана

- Цель алгоритма состоит в том, чтобы два участника могли безопасно обменяться ключом, который в дальнейшем может использоваться в каком-либо алгоритме симметричного шифрования. Сам алгоритм Диффи-Хеллмана может применяться только для обмена ключом.
- Алгоритм основан на трудности вычислений дискретных логарифмов. Дискретный логарифм определяется следующим образом. Вводится понятие примитивного корня простого числа **Q** как числа, чьи степени создают все целые от **1** до **Q–1**. Это означает, что если **A** является примитивным корнем простого числа **Q**, тогда числа

$$A \pmod{Q},\ A^2 \pmod{Q},\ \ldots,\ A^{Q-1} \pmod{Q}$$

- являются различными и состоят из целых от **1** до **Q–1** возможно с некоторыми перестановками.
- В этом случае для любого целого **B<Q** и примитивного корня **A** простого числа **Q** можно найти единственную экспоненту **X**, такую, что

$$Y = A^X \pmod{Q},\ \text{где}\ 0 \le X \le (Q-1)$$

- Экспонента **X** называется дискретным логарифмом, или индексом **Y**, по основанию **A (mod Q).** Это обозначается как

$$\text{ind}_{A,\,Q}(Y)$$

Алгоритм Диффи-Хеллмана

Теперь опишем алгоритм обмена ключей Диффи-Хеллмана.

Общеизвестные элементы

Q – Простое число

A – **A<Q и A** является примитивным корнем **Q**

Предполагается, что существуют два известных всем числа: простое число **Q** и целое **A**, которое является примитивным корнем **Q**.

Алгоритм Диффи-Хеллмана

Создание пары ключей пользователем I

Выбор случайного числа X_i - закрытый ключ

$X_i < Q$

Вычисление числа Y_i - открытый ключ

$Y_i = A^{Xi} \pmod{Q}$

Алгоритм Диффи-Хеллмана

Создание открытого ключа пользователем J

Выбор случайного числа X_j - закрытый ключ

$X_j < Q$

Вычисление числа Y_j - открытый ключ

$Y_j = A^{X_j} \pmod{Q}$

Алгоритм Диффи-Хеллмана

- Теперь предположим, что пользователи **I** и **J** хотят обменяться ключом для алгоритма симметричного шифрования. Пользователь **I** выбирает случайное число $X_i < Q$ и вычисляет $Y_i = A^{Xi} \pmod{Q}$. Аналогично пользователь **J** независимо выбирает случайное целое число $X_j < Q$ и вычисляет $Y_j = A^{Xj} \pmod{Q}$.

Алгоритм Диффи-Хеллмана

Пользователи I и J обмениваются открытыми ключами Y_i и Y_j

Каждая сторона держит значение **X** в секрете и делает значение **Y** доступным для другой стороны.

Создание общего секретного ключа пользователем I

$$K = (Y_j)^{x_i} \pmod{Q}$$

Создание общего секретного ключа пользователем J

$$K = (Y_i)^{x_j} \pmod{Q}$$

Алгоритм Диффи-Хеллмана

Теперь пользователь **I** вычисляет ключ $K = (Y_j)^{Xi} \pmod Q$, и пользователь **J** вычисляет ключ $K = (Y_i)^{Xj} \pmod Q$. В результате оба получат одно и то же значение:

$$K = (Y_j)^{Xi} \pmod Q$$
$$= (A^{Xj} \bmod Q)^{Xi} \pmod Q$$
$$= (A^{Xj})^{Xi} \pmod Q$$ по правилам модульной арифметики
$$= A^{XjXi} \pmod Q$$
$$= (A^{Xi})^{Xj} \pmod Q$$
$$= (A^{Xi} \bmod Q)^{Xj} \pmod Q$$
$$= (Y_i)^{Xj} \pmod Q$$

Таким образом, две стороны обменялись секретным ключом. Так как X_i и X_j являются закрытыми, противник может получить только следующие значения: **Q**, **A**, Y_i и Y_j. Для вычисления общего ключа атакующий должен взломать дискретный логарифм, т.е. вычислить

$$X_j = \text{ind}_{A,Q}(Y_j)$$

Алгоритм Диффи-Хеллмана

- Безопасность обмена ключа в алгоритме Диффи-Хеллмана вытекает из того факта, что, хотя относительно легко вычислить экспоненты по модулю простого числа, но очень трудно вычислить дискретные логарифмы. Для больших простых чисел задача считается неразрешимой.
- Следует заметить, что данный алгоритм уязвим для атак типа «**man-in-the-middle**». Если противник может осуществить активную атаку, т.е. имеет возможность не только перехватывать сообщения, но и заменять их другими, он может перехватить открытые ключи участников Y_i и Y_j, создать свою пару открытого и закрытого ключа $(X_{оп}, Y_{оп})$ и послать каждому из участников свой открытый ключ. После этого каждый участник вычислит ключ, который будет общим с противником, а не с другим участником. Если нет аутентификации хотя бы одной из сторон внешним по отношению к алгоритму Диффи-Хеллмана способом, то участники не смогут обнаружить подобную подмену.