*ЛАБОРАТОРНАЯ РАБОТА № 1*

# ОСНОВЫ КРИПТОСИСТЕМЫ RSA

Цель работы – изучение криптосистемы RSA, реализация алгоритмов зашифрования и расшифрования сообщений, формирования и проверки электронной цифровой подписи.

## Теоретические сведения

### 1. Краткое описание криптосистемы RSA

Криптосистема RSA, предложенная в 1977 году Р. Ривестом (Ron Rivest), А. Шамиром (Adi Shamir) и Л. Адлеманом (Leonard Adleman) и названная по первым буквам фамилий авторов, широко используется для шифрования с открытым ключом и электронной (цифровой) подписи. Криптосистема RSA является составной частью многих стандартов, а также используется во многих приложениях и протоколах, например, PGP, SSL/TLS, SSH и др.

Принцип действия криптосистемы заключается в следующем. Пусть $p$ и $q$ – два больших простых числа и пусть $n = pq$. Выбирается число $e$, $1 < e < \varphi(n) - 1$, взаимно простое со значением функции Эйлера $\varphi(n) = (p-1)(q-1)$, и вычисляется $d$ – мультипликативно обратное к $e$: $d \equiv e^{-1}(mod\ \varphi(n))$. Числа $e$ и $d$ называются открытым и закрытым показателями соответственно. Открытым ключом является пара $(n, e)$, закрытым ключом – пара $(n, d)$. Множители $p$ и $q$ известны только владельцу секретного показателя $d$ (или могут быть вычислены с помощью этого показателя).

Для зашифрования сообщения $m$ нужно вычислить шифртекст $c \equiv m^e\ (mod\ n)$, а для расшифрования – возвести шифртекст в степень $d$: $m \equiv c^d\ (mod\ n)$.

### 2. Применение криптосистемы RSA для шифрования

Одна операция зашифрования в системе RSA позволяет зашифровать сообщение, представленное числом $m$, где $0 \le m < n$. Текст более длинных сообщений необходимо разбивать на блоки. То же

справедливо и для файлов: преобразование из потока байтов в число для зашифрования подразумевает разделение файла на блоки байтов, преобразование каждого такого блока в число и зашифрование каждого блока в отдельности.

При преобразовании сообщения (последовательности байтов) в число можно использовать два варианта представления. В первом варианте полагается, что первые байты сообщения определяют старшие цифры числа (Most Significant Bytes first, MSB), а во втором – первые байты сообщения являются младшими цифрами числа (Least Significant Bytes first, LSB).

Как и все несимметричные алгоритмы, алгоритм шифрования RSA требует значительных вычислительных ресурсов для выполнения каждой операции зашифрования. Поэтому на практике несимметричные алгоритмы шифрования часто используют совместно с симметричными алгоритмами.

Пусть используется симметричный блочный шифр $C$ с размером блока $l_b$ и длиной ключа $l_k$. Для передачи каждого нового сообщения $m$ генерируется случайный ключ $k$ шифра $C$. Сгенерированный ключ представляется в виде числа и шифруется алгоритмом RSA. Основное сообщение разбивается на блоки длиной $l_b$ и шифруется на ключе $k$ симметричным шифром $C$.

Получателю такого составного сообщения передаются зашифрованный при помощи RSA ключ симметричного алгоритма $c = RSA_e(k)$ и шифртекст $C_k(m)$ основного сообщения$C_k(m)$. Получатель с помощью закрытого ключа $d$ расшифровывает ключ $k$ симметричного алгоритма, а затем основное сообщение $m$.

Размер ключа $l_k$ обычно значительно меньше длины числа $n$ в байтах, поэтому для шифрования такого ключа требуется выполнение только одной операции RSA.

Применение такого подхода, помимо значительного увеличения скорости передачи длинных сообщений, обладает еще одним преимуществом: упрощается передача широковещательных сообщений. Для нескольких получателей ключ, на котором зашифровано сообщение $m$, передается зашифрованным на различных ключах зашифрования: $RSA_{e_1}(k)$, $RSA_{e_2}(k)$, …,$RSA_{e_n}(k)$. Получатели такого сообщения,

каждый на своем закрытом ключе, расшифровывают промежуточный ключ $k$ и получают доступ к сообщению.

Вместе с зашифрованным промежуточным ключом необходимо передавать также значение или идентификатор ключа шифрования RSA, чтобы каждый из получателей смог определить, какой именно блок из множества переданных необходимо расшифровать для получения промежуточного ключа.

## 3. Применение криптосистемы RSA для подписи

До появления криптографии с открытым ключом целостность и подлинность документов в условиях взаимного недоверия контролировались, как правило, по материальным признакам. Документы имели конкретное физическое воплощение, например, были напечатаны на гербовой бумаге или заверены печатью, но с ростом информатизации стали терять уникальные физические качества.

Использование симметричных криптографических алгоритмов не позволяет установить подлинность и целостность электронного документа в условиях взаимного недоверия. Например, если получатель сообщения утверждает, что получил от отправителя некоторое заверенное сообщение, а отправитель это отрицает, то один из них лжет. Установить, кто именно говорит неправду, затруднительно, так как оба они обладают равными возможностями в части формирования имитозащищенных сообщений. Осуществить аутентификацию данных и их источника в этом случае позволяет электронная подпись.

Электронная подпись в цифровых документах играет ту же роль, что и подпись, поставленная от руки в бумажных документах: это данные, присоединяемые к передаваемому сообщению и подтверждающие, что автор подписи (отправитель) составил или заверил данное сообщение. Получатель сообщения или третья сторона с помощью электронной подписи может проверить, что автором сообщения является именно владелец подписи и что в процессе передачи не была нарушена целостность данных.

Протокол электронной подписи предполагает два этапа: формирование и проверку. Часто способы формирования и проверки электронной подписи допускают некоторую свободу действий, которую

нет смысла строго регламентировать. Это позволяет говорить о схеме подписи.

Таким образом, при разработке протокола подписи нужно:

– создать пару сообщение/подпись так, чтобы ее невозможно было подделать;

– уметь проверить, что подпись действительно принадлежит указанному владельцу.

Кроме того, подпись должна быть построена так, чтобы отправитель, подписавший сообщение, не смог затем отрицать перед получателем или третьей стороной факт подписания, утверждая, что подпись подделана.

Обычно, говоря о схеме электронной подписи, имеют в виду следующую ситуацию:

– отправитель знает содержание сообщения, которое он подписывает;

– получатель, зная открытый ключ проверки подписи, может проверить правильность подписи в любое время без какого-либо разрешения или участия отправителя;

– безопасность схемы подписи (то есть трудность подделки, невозможность отказа от подписи и т. д.) обеспечивается положениями теории сложности.

При формировании электронной подписи по классической схеме отправитель сообщения сначала применяет к исходному тексту хэш-функцию, а затем вычисляет подпись с помощью закрытого ключа формирования подписи.

Получатель сообщения выполняет проверку подписи: сначала вычисляет хэш-образ полученного сообщения с помощью той же самой хэш-функции, которую использовал отправитель, а затем с помощью открытого ключа проверяет, соответствует ли этот хэш-образ полученной электронной подписи.

## 4. Схема подписи RSA

**Протокол 1.1.** Схема подписи RSA

*Вход отправителя*. Составное число $n$, закрытый показатель $d$.

*Вход получателя*. Составное число $n$, открытый показатель $e$.

*Результат*. Формирование и проверка подписи.

Для формирования подписи для сообщения $m$ отправитель выполняет следующие действия.

1. Вычисляет хэш-образ $r \leftarrow h(m)$ сообщения $m$.
2. Зашифровывает число $r$ на своем закрытом ключе $d$, то есть вычисляет экспоненту $s \leftarrow r^d \ (mod\ n)$.

Подписью для сообщения $m$ является число $s$.

Для проверки подписи получатель выполняет следующие действия.

1. Расшифровывает подпись $s$ на открытом ключе $e$ отправителя, то есть вычисляет $t \leftarrow s^e \ (mod\ n)$, восстанавливая предполагаемый хэш-образ $t$ сообщения $m$.
2. Вычисляет хэш-образ $r \leftarrow h(m)$ сообщения $m$.
3. Проверяет равенство $r = t$. Если оно выполняется, то подпись принимается, в противном случае – подпись неверна. ■

Схема подписи RSA может быть рассмотрена как категория, объектами которой являются пары сообщение/подпись, а морфизмами – отображения таких пар, удовлетворяющие проверочным соотношениям. Если $(m, s)$ – подлинное подписанное сообщение, то пара $(m', s')$ тоже будет правильно подписанным сообщением, если $h(m') \equiv h^k(m)(mod\ n)$ и $s' \equiv s^k(mod\ n)$ для некоторого целого $k$. Новая пара$(m', s')$ может быть создана следующим алгоритмом: выбрать сообщение $m'$, вычислить $h(m')$, найти логарифм $h(m')$ по основанию $h(m)$ и вычислить $s'$. Можно наоборот, сначала найти $r' \equiv h^k(m)(mod\ n)$ для известного $k$, а затем подобрать $m'$ так, чтобы выполнялось равенство $h(m') = r'$. Поэтому стойкость подписи RSA не может превышать сложности вычисления логарифма в группе $(\mathbb{Z}/n\mathbb{Z})^*$ или сложности обращения хэш-функции.

Если нарушитель составляет текст для подписи, то он может подготовить коллизию $(m, m')$ такую, что $h(m) = h(m')$, дать на подпись сообщение $m$, а затем подменить подписанное сообщение $m$ сообщением $m'$. В этом случае стойкость схемы подписи не может превышать сложности вычисления коллизий хэш-функции. Приведенные рассуждения показывают, что необходимо согласовывать сложности

разложения числа $n$ на множители, логарифмирования и нахождения коллизий.

## 5. Представление ключей RSA

Открытый и закрытый ключи RSA обладают определенной структурой, например, нельзя поменять местами модуль и экспоненту. Кроме того, длины компонент могут сильно различаться. Так, длины ключей RSA в различных системах могут отличаться в разы. Стойкость криптосистемы со временем падает, и для компенсации необходимо постоянно увеличивать длину ключей. Поэтому к методу представления ключа как массива байтов предъявляется ряд требований, а именно:

– информация должна храниться в определенном известном формате;

– не должно быть ограничений по максимальной возможной длине компонент структуры;

– не должно быть выравнивания остальных компонент по наиболее длинной.

Этим требованиям удовлетворяет представление ключа в виде ASN.1-структуры. Такое представление предназначено для хранения и передачи структурированных данных. Подробное описание типов данных в ASN.1-структуре, форматов зашифрованного файла и файла подписи представлено в *Приложении А*.

## Контрольные вопросы

1. Какие задачи положены в основу безопасности системы RSA?
2. Показать, что схема RSA работает корректно для любого сообщения $m \in \mathbb{Z}/n\mathbb{Z}$.
3. Доказать, что задача разложения числа $n$ на множители и задача вычисления функции Эйлера $\varphi(n)$ полиномиально эквивалентны.
4. Показать, что схема RSA обладает свойством гомоморфности относительно операции умножения.

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, которая:

а) выполняет зашифрование и расшифрование файла с использованием алгоритма RSA. В режиме зашифрования программа должна принимать на вход файл сообщения (произвольного формата), открытый ключ RSA, ключ симметричного алгоритма; на выходе − возвращать зашифрованный файл, заголовок которого соответствует нотации ASN.1, описанной в *Приложении А*. Зашифрование файла необходимо производить с использованием вспомогательного симметричного алгоритма AES-256 в режиме CBC, реализация симметричного алгоритма не требуется. Ключ шифрования симметричного алгоритма – случайный или задан в программе. Случайный ключ шифрования алгоритма AES (32 байта) необходимо представить как число для шифрования алгоритмом RSA, порядок байтов – MSB. Старшие неиспользуемые цифры (байты) числа следует считать нулевыми. В режиме расшифрования программа **П-1** должна принимать на вход зашифрованный файл, закрытый ключ RSA; на выходе – возвращать расшифрованное сообщение;

б) выполняет формирование и проверку электронной подписи с использованием алгоритма RSA. При реализации алгоритма электронной подписи рекомендуется использовать хэш-алгоритм SHA-256. Поскольку хэш-образ, вычисленный по алгоритму SHA-256, может быть длиннее модуля RSA, при проверке подписи следует сравнивать значение $h(m)(mod\ n)$ с подписью $s$ сообщения $m$. В режиме формирования подписи программа **П-1** должна принимать на вход файл, который необходимо подписать, ключ подписи RSA; на выходе – возвращать отдельный файл подписи. В режиме проверки подписи программа должна принимать на вход сообщение и файл подписи; на выходе – возвращать результат «Подпись принимается» или «Подпись неверна».

## Содержание отчета

1. Формулировка задания.

2. Выполненная работа:

а) использованные параметры криптосистемы RSA;

б) сгенерированный ключ для алгоритма AES-256;

в) содержание зашифрованного файла в шестнадцатеричном виде (при значительном размере файла – первые 80–100 байт);

г) файл с подписью RSA.

3. Ответы на контрольные вопросы.

4. Выводы по работе, где сделать предположения о возможных уязвимостях разработанного программного обеспечения и о путях их устранения.

5. Листинг программы.

*ЛАБОРАТОРНАЯ РАБОТА № 2*

# АТАКИ НА КРИПТОСИСТЕМУ RSA, ИСПОЛЬЗУЮЩИЕ ДАННЫЕ О ПОКАЗАТЕЛЯХ

Цель работы – изучение уязвимостей криптосистемы RSA по отношению к атакам, использующим показатели, обладающие определенными свойствами, генерация параметров криптосистемы RSA.

## Теоретические сведения

Как показывает практика, правильный выбор множителей $p$ и $q$ не всегда гарантирует стойкость криптосистемы. Существует множество атак, нарушающих безопасность криптосистемы RSA без разложения числа $n$ на множители. Возможность подобных атак, как показано в работе Д. Боне «Twenty years of attacks on the RSA cryptosystem», обусловлена использованием общего модуля $n$ несколькими пользователями, малых значений открытого и закрытого показателей и т.п. Наличие таких атак накладывает значительные ограничения на выбор параметров криптосистем на основе алгоритма RSA.

### 1. Случай общего модуля

Если в системе RSA у пользователей $A$ и $B$ общее составное число $n$, но различные открытые $e_A, e_B$ и закрытые $d_A, d_B$ показатели, где $e_A d_A \equiv e_B d_B \equiv 1 (mod\ \varphi(n))$, то каждый пользователь может разложить число $n$ на множители и тем самым узнать закрытый ключ другого пользователя. Для этого нарушителю (например, пользователю $B$) достаточно найти квадратный корень из единицы, отличный от $\pm 1$. Если $t^2 \equiv 1\ (mod\ n), t \neq \pm 1\ (mod\ n)$, то выполняется сравнение $t^2 - 1 = (t+1)(t-1) \equiv 0\ (mod\ n)$ и числа $t+1, t-1$ имеют нетривиальные общие делители с $n$. Разложение можно выполнить по аналогии с псевдопростым тестом Миллера–Рабина следующим алгоритмом.

**Алгоритм 2.1.** Разложение составного числа на множители по известным показателям RSA

*Вход*. Число $n$, показатели $e_B, d_B$ такие, что $e_B d_B \equiv 1\ (mod\ \varphi(n))$, открытый показатель $e_A$ другого пользователя.

*Выход*. Делители $p$ и $q$ числа $n$, закрытый показатель $d_A$ другого пользователя.

1. Представить разность $e_B\ d_B - 1$ в виде $2^f s$, где $s$ – нечетное число.
2. Выбрать случайное $a \in \mathbb{Z}/n\mathbb{Z}$ и вычислить $b \leftarrow a^s (mod\ n)$.
3. Вычислять $b^{2^0} \equiv b\ (mod\ n)$, $b^{2^1} \equiv (b^{2^0})^2\ (mod\ n)$, $b^{2^2} \equiv (b^{2^1})^2\ (mod\ n)$, … до получения $l$ такого, что $b^{2^l} \equiv 1\ (mod\ n)$. Если $b^{2^{l-1}} \equiv -1\ (mod\ n)$, то вернуться на шаг 2, иначе положить $t \leftarrow b^{2^{l-1}}\ (mod\ n)$.
4. Положить $p \leftarrow$ НОД$(t+1, n)$, $q \leftarrow$ НОД$(t-1, n)$.
5. Вычислить $\varphi(n) = (p-1)(q-1)$ и найти закрытый ключ другого пользователя: $d_A \equiv e_A^{-1} (mod\ \varphi(n))$.
6. Результат: $(p, q), d_A$. ■

## 2. Случай малого закрытого показателя

Атака Винера на схему RSA основана на предположении о том, что значение закрытого показателя $d$ достаточно мало. Пусть $n = pq$, $q < p < 2q$, $(n, e)$ − открытый ключ, $(n, d)$ − закрытый ключ схемы RSA. Тогда если $d < \frac{1}{3}\sqrt[4]{n}$, то значение закрытого показателя $d$ может быть вычислено за полиномиальное от $n$ время.

Для доказательства этого предположения используется математический аппарат непрерывных дробей, который позволяет найти приближение к некоторому вещественному числу $\alpha$ с помощью подходящих дробей. Пусть $\frac{P}{Q}$ − несократимая рациональная дробь, и выполняется условие: $\left|\alpha - \frac{P}{Q}\right| < \frac{1}{2Q^2}$, тогда $\frac{P}{Q}$ представляет собой подходящую дробь к числу $\alpha$.

Открытый и закрытый показатели связаны соотношением $ed \equiv 1\ (mod\ \ \varphi(n))$, т.е. существует такое $k \in \mathbb{Z}$, что $ed = 1\ + k\varphi(n)$, или

$|ed - k\varphi(n)| = 1$. При делении последнего равенства на $d\varphi(n)$ получаем:

$$\left|\frac{e}{\varphi(n)} - \frac{k}{d}\right| = \frac{1}{d\varphi(n)}.$$

Так как значение $\varphi(n)$ неизвестно, вместо $\frac{e}{\varphi(n)}$ можно использовать соотношение известных значений $\frac{e}{n}$ ввиду относительной близости чисел $n$ и $\varphi(n)$ друг к другу:

$$|n - \varphi(n)| = n - pq + p + q - 1 =$$
$$= p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}.$$

Тогда при $k < d$ и $d < \frac{1}{3}\sqrt[4]{n}$ выполняется неравенство $3k < 3d < \sqrt[4]{n}$, откуда следует:

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \left|\frac{\overbrace{ed - k\varphi(n)}^{=1} - kn + k\varphi(n)}{nd}\right| = \left|\frac{1 + k(\varphi(n) - n)}{nd}\right| < \frac{3k\sqrt{n}}{nd} \leq \frac{1}{3d^2} < \frac{1}{2d^2}.$$

Таким образом, подходящая дробь $\frac{k}{d}$ со знаменателем, близким к $\sqrt[4]{n}$, является хорошим приближением к $\frac{e}{n}$. Действуя таким образом, можно найти закрытый показатель $d$, если его длина достаточно мала.

**Алгоритм 2.2.** Атака Винера на криптосистему RSA

*Вход*. Число $n$, показатель $e$.

*Выход*. Закрытый показатель $d$.

1. Представить $\frac{e}{n}$ в виде обыкновенной непрерывной дроби $[0; a_1, a_2, \ldots, a_l]$, где $l \approx log_2\, n$.
2. Для $i = 1, \ldots, l$ выполнить следующие действия.
   2.1. Вычислить i-ю подходящую дробь $\frac{P_i}{Q_i}$ к $\frac{e}{n}$:
   $\frac{P_i}{Q_i} = \frac{a_i P_{i-1} + P_{i-2}}{a_i Q_{i-1} + Q_{i-2}}$ $\frac{P_i}{Q_i} = \frac{a_i P_{i-1} + P_{i-2}}{a_i Q_{i-1} + Q_{i-2}}$, где $P_{-1} = 1, Q_{-1} = 0,\ P_0 = 0,$ $Q_0 = 1$.
   2.2. Для произвольного сообщения $m \in \mathbb{Z}/n\mathbb{Z}$ проверить выполнение сравнения $(m^e)^{Q_i} \stackrel{?}{\equiv} m (mod\, n)$. Если сравнение выполняется, то положить $d \leftarrow Q_i$ и перейти на шаг 3.
3. Результат: $d$. ■

### 3. Случай специальных открытых показателей

Каждый $i$-й пользователь системы связи с шифрованием по схеме RSA должен иметь свое персональное число $n_i = p_i q_i$. Для ускорения процесса шифрования иногда используются малые открытые показатели $e_i$, причем они могут быть одинаковыми, например, 3 или 5. В этом случае, даже если числа $n_i$ различны, можно выполнить бесключевое дешифрование сообщения. Предположим, что отправитель посылает одно и то же сообщение $m$ трем получателям, открытые ключи которых равны $(n_i, e_i)$, причем $e_i = 3$ и $m < n_i$. Обозначим $x = m^3$. Тогда выполняются сравнения $c_i \equiv m^3 \pmod{n_i}$, $x \equiv c_i \pmod{n_i}$, и значение $x$ как целое число может быть восстановлено по китайской теореме об остатках по известным $c_i$ и взаимно простым модулям $n_i$. Для нахождения сообщения $m$ нужно извлечь кубический корень из $x$ в кольце целых чисел. Эта операция имеет полиномиальную сложность. Очевидно, то же справедливо и в случае других малых одинаковых открытых показателей.

**Алгоритм 2.3.** Бесключевое дешифрование широковещательного сообщения в случае малого общего показателя $e$

*Вход*. Открытые ключи нескольких пользователей:$(e, n_1), (e, n_2), \ldots, (e, n_z)$, шифртексты широковещательного сообщения: $c_1, c_2, \ldots, c_z$.

*Выход*. Сообщение $m$.

1. По известным шифртекстам $c_1, c_2$,...,$c_z$ и модулям $n_1, n_2, \ldots, n_z$ по китайской теореме об остатках восстановить число $x = m^e$.
2. Вычислить $m = \sqrt[e]{x} \in \mathbb{Z}$.
3. Результат: $m$. ■

Еще одна атака на схему шифрования возможна, если мал порядок элемента $e$ в группе $(\mathbb{Z}/\varphi(n)\mathbb{Z})^*$. Для нахождения сообщения $m$ по шифртексту $c = m^e \ (mod \ \ n)$ необходимо выполнить последовательное зашифрование шифртекста: $c_1 = c^e \ (mod\, n)$, $c_2 \equiv c_1^e (mod\, n)$

и т. д. до получения шифртекста $c$, тогда предыдущее значение и будет открытым текстом.

**Алгоритм 2.4.** Бесключевое дешифрование сообщения в случае малого порядка элемента $e$ в группе $(\mathbb{Z}/\varphi(n)\mathbb{Z})^*$

*Вход*. Составное число $n$, шифртекст $c$, открытый показатель $e$.

*Выход*. Открытый текст $m$.

1. Для $i = 1, 2, \ldots$ выполнить следующие действия.

1.1. Вычислить $c_i \equiv c_{i-1}^e \equiv c_0^{e^i} \ (mod\ n)$, где $c_0 \leftarrow c$.

1.2. Проверить выполнение сравнения $c_i \stackrel{?}{\equiv} c \pmod n$. Если сравнение выполняется, то положить $m \leftarrow c_{i-1} \equiv c_0^{e^{ord(e)-1}} (mod\ n)$ и перейти на шаг 3.

2. Результат: $m$. ■

Отметим, что порядок элемента $e$ в группе $(\mathbb{Z}/\varphi(n)\mathbb{Z})^*$ мал лишь в том случае, если он мал в обеих группах $\mathbb{F}_p^*$ и $\mathbb{F}_q^*$. Если этот порядок мал хотя бы в одной из указанных групп, то можно разложить число $n$ следующим образом. Сначала сгенерировать случайное $a_0 \in (\mathbb{Z}/n\mathbb{Z})^*$. Затем для $i \geq 1$ вычислять последовательно значения $a_i \equiv a_{i-1}^e (mod\ n)$ и проверять с помощью алгоритма Евклида условие НОД$(a_i - a_0, n) > 1$. Если это условие будет выполнено, то указанный наибольший общий делитель является нетривиальным делителем числа $n$. Для исключения этой атаки числа $p$ и $q$ выбирают так, чтобы порядки групп $\mathbb{F}_p^*$ и $\mathbb{F}_q^*$ имели большие простые делители.

В случае малых открытых показателей можно не только осуществить бесключевое дешифрование, но и вскрыть закрытый показатель $d$. Для этого достаточно знать четверть двоичных разрядов одного из делителей числа $n$.

## Контрольные вопросы

1. При каком условии возможна атака Винера?
2. Перечислите методы противодействия атаке Винера.
3. Почему числа $p$ и $q$ необходимо выбирать так, чтобы $p-1, q-1$ имели большие простые делители?

4. Предложите способы защиты от атаки бесключевого дешифрования широковещательных сообщений без отказа от использования общих малых закрытых показателей.

## Порядок выполнения работы

1. Получить у преподавателя вариант задания.
2. Разработать программу **П-1**, которая реализует разложение составного числа на множители по известным показателям RSA и вычисляет закрытый ключ другого пользователя в случае общего модуля $n$. Программа **П-1** должна принимать на вход открытый и закрытый ключ RSA, а также открытый ключ RSA другого пользователя; на выходе – возвращать множители модуля RSA, закрытый ключ другого пользователя.
3. Разработать программу **П-2**, которая реализует атаку Винера на криптосистему RSA. Программа **П-2** должна принимать на вход открытый ключ RSA: число $n$ и показатель $e$; на выходе – возвращать закрытый показатель $d$.
4. Разработать программу **П-3**, которая, в соответствии с вариантом задания, реализует одну из следующих атак:

− бесключевое дешифрование широковещательного сообщения в случае малого общего показателя $e$. Для восстановления числа $m^e$ по китайской теореме об остатках и нахождения корня степени $e$ допускается использовать готовые процедуры. Тело сообщения – файл, зашифрованный алгоритмом AES-256 в режиме CBC. При длине исходного файла, не кратной 16 байтам, открытый текст дополняется символами с кодом 0x03 так, чтобы длина сообщения (файла) перед зашифрованием была кратна 16 байтам. Изначальная длина доступна в элементе данных «Длина сообщения» заголовка. Удаление выравнивания не обязательно. Программа **П-3** должна принимать на вход открытые ключи нескольких пользователей, а также шифртексты широковещательного сообщения; на выходе – возвращать восстановленный открытый текст;

− бесключевое дешифрование сообщения в случае малого порядка элемента $e$ в группе $(\mathbb{Z}/\varphi(n)\mathbb{Z})^*$.. Программа **П-3** должна при-

нимать на вход открытый ключ RSA, шифртекст; на выходе – возвращать восстановленный открытый текст.

5. Разработать программу **П-4**, которая осуществляет генерацию параметров криптосистемы RSA. Разработка процедур генерации случайных чисел и проверки числа на простоту не требуется. Генерация ключей должна обеспечивать выработку безопасных параметров криптосистемы с учетом произведенных атак. Программа **П-4** должна принимать на вход битовую длину модуля RSA; на выходе – возвращать параметры криптосистемы: $n, e, d$. Выполнить анализ сгенерированных параметров с помощью инструмента CrypTool. Описание утилиты представлено в *Приложении Б*.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) описание осуществленных атак;

б) параметры схемы RSA, используемые при атаках;

в) содержимое дешифрованных файлов и его тип, найденные значения закрытого ключа;

г) описание разработанной процедуры генерации параметров и ключей системы RSA;

д) пример сгенерированных ключей и параметров криптосистемы как последовательности модуля и экспоненты;

е) результаты тестирования сгенерированных параметров криптосистемы с помощью программы CrypTool.

3. Ответы на контрольные вопросы.
4. Выводы по работе.
5. Листинги программ.

*ЛАБОРАТОРНАЯ РАБОТА № 3*

# АТАКА НА КРИПТОСИСТЕМУ RSA ПРИ ШИФРОВАНИИ КОРОТКИХ СООБЩЕНИЙ

Цель работы – изучение атак на криптосистему RSA, связанных с использованием малого открытого показателя при шифровании коротких сообщений, реализация схемы дополнения RSA-OAEP.

## Теоретические сведения

Среди множества атак на криптосистему RSA выделяют класс атак, связанных с использованием малого открытого показателя (например, $e = 3$). Такой выбор ключа чаще всего обусловлен необходимостью повышения производительности операций зашифрования сообщений и проверки подписи, однако существенно снижает криптографическую стойкость системы RSA.

При шифровании небольшим открытым ключом короткого сообщения $m^e < n$ восстановление исходного $m$ по известному шифртексту $c$ возможно путем извлечения корня степени $e$ из $c$ как целого числа. Для защиты от подобной атаки рекомендуется выполнять дополнение сообщения до фиксированный длины. Самый простой вариант дополнения подразумевает добавление случайных младших битов к сообщению. Однако данная мера не позволяет обеспечить достаточный уровень безопасности.

### 1. Случай связанных между собой сообщений

Если два сообщения $m_1$, $m_2$ не одинаковы, но связаны аффинным соотношением: $m_2 \equiv \alpha m_1 + \beta \ (mod\ n)$, где $\alpha$ и $\beta$ известны, то по шифртекстам $c_1$, $c_2$ можно восстановить оба сообщения. Например, для $e = 3$ сообщение $m_1$ восстанавливается по формуле:

$$\frac{\beta(c_2+2\alpha^3 c_1-\beta^3)}{\alpha(c_2-\alpha^3 c_1+2\beta^3)} = \frac{3\alpha^3\beta{m_1}^3+3\alpha^2\beta^2{m_1}^2+3\alpha\beta^3 m_1}{3\alpha^3\beta{m_1}^2+3\alpha^2\beta^2 m_1+3\alpha\beta^3} \equiv m_1 (mod\, n).$$

Сообщение $m_2$ выражается через $m_1$ и коэффициенты $\alpha, \beta$. Для произвольного показателя $e$ имеют место соотношения

$$m_1^e - c_1 \equiv 0 \ (mod\, n),\ (\alpha m_1 + \beta)^e - c_2 \equiv 0 \ (mod\, n).$$

Обозначим неизвестное сообщение $m_1$ через переменную $z$. Тогда, переходя к кольцу полиномов $(\mathbb{Z}/n\mathbb{Z})[z]$, можно записать:

$$\text{НОД}(z^e - c_1, (\alpha z + \beta)^e - c_2) = z - m_1 \in (\mathbb{Z}/n\mathbb{Z})[z].$$

Вычислив этот наибольший общий делитель, можно определить $m_1$.

**Алгоритм 3.1.** Нахождение двух открытых текстов по двум шифртекстам и параметрам аффинного преобразования

*Вход*. Составное число $n$, шифртексты $c_1$, $c_2$, параметры $\alpha$, $\beta$, открытый показатель $e$.

*Выход*. Открытые тексты $m_1, m_2$.

1. Вычислить $m_1$ как свободный член линейного полинома $z - m_1 = \text{НОД}(z^e - c_1, (\alpha z + \beta)^e - c_2)$ в кольце $(\mathbb{Z}/n\mathbb{Z})[z]$.
2. Положить $m_2 \leftarrow \alpha m_1 + \beta \ (mod\, n)$.
3. Результат: $(m_1, m_2)$. ■

Алгоритм 3.1 допускает обобщение на случай, когда $m_2$ задается произвольным полиномом от $m_1$ над $\mathbb{Z}/n\mathbb{Z}$, то есть $m_2 = f(z)|_{z=m_1}$. Тогда $\text{НОД}(z^e - c_1, f(z)^e - c_2) = z - m_1$ в кольце $(\mathbb{Z}/n\mathbb{Z})[z]$. Если $m_1$ и $m_2$ задаются полиномами от $m$ над кольцом$\mathbb{Z}/n\mathbb{Z}$:

$$m_1 = f_1(z)|_{z=m},\ \ m_2 = f_2(z)|_{z=m},$$

и известны шифртексты $c_1$, $c_2$, то

$$\text{НОД}(f_1(z)^e - c_1, f_2(z)^e - c_2) = z - m$$

в кольце $(\mathbb{Z}/n\mathbb{Z})[z]$.

**Пример 3.1.** Восстановление пары сообщений без разложения числа $n$

Пусть

$$n = 81089,\ c_1 = 32205,\ c_2 = 38069,\ e = 17,\ \alpha = 1,\ \beta = 1.$$

Вычисляем $\text{НОД}((z+1)^{17} - 38069, z^{17} - 32205)$ в кольце $(\mathbb{Z}/n\mathbb{Z})[z]$. Получаем последовательность вычетов для алгоритма Евклида (с учетом нормирования полиномов по старшему коэффициенту):

$$z^{16} + 8z^{15} + 40z^{14} + 140z^{13} + 364z^{12} + 728z^{11} +$$

$$+ 1144z^{10} + 1430z^9 + 1430z^8 + 1144z^7 + 728z^6 +$$
$$+ 364z^5 + 140z^4 + 40z^3 + 8z^2 + z + 9195;$$
$$z^{15} + 40552z^{14} + 40576z^{13} + 91z^{12} + 195z^{11} +$$
$$+ 20594z^{10} + 47719z^9 + 429z^8 + 351z^7 + 40772z^6 +$$
$$+ 40660z^5 + 45z^4 + 13z^3 + 30411z^2 + 36783z + 72676;$$
$$\ldots$$
$$z^2 + 79994z + 24207;$$
$$z + 68744 = z - 12345,$$

то есть $m_1 = 12345$.

Второе сообщение: $m_2 = m_1 + 1 = 12346$. ■

Операция вычисления наибольшего общего делителя полиномов степени $e$ над кольцом $\mathbb{Z}/n\mathbb{Z}$ предусматривает выполнение на каждой итерации следующих действий:

– одну операцию обращения по модулю $n$;

– $O(e)$ операций умножения чисел по модулю $n$;

– $O(e)$ операций сложения чисел по модулю $n$.

Число итераций равно $O(e)$.

При использовании умножения «в столбик» и арифметики Монтгомери (см. работу № 4) сложность алгоритма 3.1**3.** равна $O(e^2(log\, n)^2)$. При использовании умножения с помощью быстрого преобразования Фурье сложность алгоритма равна $O(e^2(log\, n)(loglog\, n))$.

Для того чтобы стойкость к методу дешифрования на основе частично известных открытых текстов была не хуже, чем сложность разложения, открытый показатель $e$ должен быть достаточно велик. Минимально возможные значения показателя $e$ для различных размеров задачи разложения, для которых сложность указанной атаки равна сложности разложения, приведены в таблице 3.1.

Таблица 3.1 – Минимальный размер открытого показателя $e_{min}$ в системе RSA, при котором атака по алгоритму 3.1 неэффективна

| $log_2 n$ | 512 | 768 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|
| $e_{min}$ | $4{\cdot}10^8$ | $2{\cdot}10^{11}$ | $2{\cdot}10^{13}$ | $3{\cdot}10^{20}$ | $10^{31}$ | $3{\cdot}10^{46}$ |

Сложность разложения оценивалась по отношению к методу решета числового поля. Сложность атаки на основе алгоритма Евкли-

да в кольце $(\mathbb{Z}/n\mathbb{Z})[z]$ оценивалась по отношению к умножению с помощью быстрого преобразования Фурье.

На практике случаи использования отправителем связанных между собой сообщений с известным аффинным преобразованием встречаются достаточно редко. Усилением указанной атаки является вариант, возникающий при дополнении коротких сообщений случайными битами.

## 2. Случай коротких сообщений

Пусть отправитель посылает сообщение $m$, дополненное следующим образом: $m_1 = 2^\mu m + r_1, 0 \le r_1 < 2^\mu, \mu = \left\lfloor \frac{log_2 n}{e^2} \right\rfloor$, и зашифрованное на открытом ключе $e$. Нарушитель перехватывает шифртекст $c_1 \equiv (2^\mu m + r_1)^e (mod\ n)$, и сообщение не будет доставлено получателю. Не получив ответа, отправитель опять выполняет дополнение того же сообщения и направляет зашифрованное сообщение $m_2 = 2^\mu m + r_2, 0 \le r_2 < 2^\mu$, получателю. Нарушитель перехватывает второе сообщение $c_2 \equiv (2^\mu m + r_2)^e (mod\ n)$ и в результате имеет два шифртекста, соответствующих одному и тому же сообщению, но дополненному различными случайными битами. Тогда по открытому ключу $(n, e)$ и шифртекстам $c_1, c_2$, нарушитель (не зная $r_1, r_2$) может восстановить исходный текст $m$.

Рассмотрим два полинома: $g_1(x, y) = x^e - c_1$, $g_2(x, y) = (x + y)^e - c_2$. Выразив $2^\mu m = m_1 - r_1$, $m_2 = m_1 + (r_2 - r_1)$, можно определить $y = r_2 - r_1$. Тогда полиномы $g_1$ и $g_2$ будут иметь общий корень $x = m_1$. Другими словами, $\Delta = r_2 - r_1$ – это корень результанта $h(y) = Re\, s_x\, (g_1, g_2)$, представляющего собой произведение всех возможных разностей корней полиномов $g_1$ и $g_2$. Результант полиномов

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0,$$
$$g(x) = b_m x^m + b_{m-1} x^{m-1} + \ldots + b_1 x + b_0$$

можно вычислить через определитель матрицы Сильвестра

$$S_{f,g} = \begin{vmatrix} a_n & a_{n-1} & \dots & a_0 & 0 & \dots & 0 & 0 \\ 0 & a_n & a_{n-1} & \dots & a_0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_n & a_{n-1} & \dots & a_1 & a_0 \\ b_m & b_{m-1} & \dots & \dots & b_0 & \dots & 0 & 0 \\ 0 & b_m & b_{m-1} & \dots & \dots & b_0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & b_m & b_{m-1} & \dots & \dots & b_1 & b_0 \end{vmatrix}.$$

Степень $deg\ h(y) \le e^2$, более того, $|\Delta| < 2^{\mu} < n^{1/e^2}$. Следовательно, $\Delta$ является малым корнем многочлена $h$ по модулю $n$ и может быть найден по теореме Копперсмита, согласно которой, для нормированного полинома $h$ в кольце $\mathbb{Z}/n\mathbb{Z}$ за полиномиальное время могут быть найдены все корни $x_0$ такие, что $|x_0| < \sqrt[D]{n}, D = deg\ h$ и $h(x_0) \equiv 0(mod\ n)$. Тогда при известном $\Delta$ можно найти $m_1, m_2$ с помощью алгоритма 3.1, положив $\alpha = 1, \beta = r_2 - r_1$, а затем восстановить значение $m$.

**Алгоритм 3.2.** Нахождение открытого текста по двум известным шифртекстам в случае дополнения случайными битами

*Вход*. Составное число $n$, шифртексты $c_1$, $c_2$, открытый показатель $e$.

*Выход*. Открытый текст $m$.

1. Положить $g_1(x, y) = x^e - c_1$, $g_2(x, y) = (x + y)^e - c_2$.
2. Вычислить корень $\Delta$ полинома $h(y) = Res_x(g_1, g_2)$.
3. Положить $\alpha = 1, \beta = \Delta$ и найти $m_1, m_2$ согласно алгоритму 3.1.
4. Восстановить значение $m$.
5. Результат: $m$. ■

Таким образом, простейший вариант дополнения коротких сообщений снижает криптографическую стойкость криптосистемы RSA. Поэтому на практике не рекомендуется использовать малые открытые показатели, а также схемы дополнений на основе аффинного преобразования сообщений.

## 3. Схема дополнения RSA-OAEP

RSA-OAEP (Optimal Asymmetric Encryption Padding, оптимизированное асимметричное дополнение шифрования) представляет собой схему дополнения для зашифрования коротких сообщений, например, ключа симметричного алгоритма. Впервые данное преобразование было предложено М. Белларе и Ф. Рогавэй в 1994 году для защиты от атак на основе подобранных шифртекстов.

Пусть длина модуля RSA $n$ равна $k$ бит, $k_0, k_1$ − фиксированные числа (например, $k_0, k_1 > 128$). В рамках схемы приняты следующие обозначения:

- $m$ – сообщение, состоящее из $k - k_0 - k_1$ бит;
- $G: \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$ – хэш-функция;
- $H: \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$ – хэш-функция;
- $r$ – случайная строка длины $k_0$ бит.

**Протокол 3.1.** Схема шифрования RSA-OAEP

*Вход отправителя*. Сообщение $m$, открытый ключ RSA $(n, e)$, хэш-функции $G$ и $H$.

*Вход получателя*. Закрытый показатель RSA $d$, хэш-функции $G$ и $H$.

*Результат*. Зашифрование и расшифрование сообщения $m$.

Для зашифрования сообщения $m$ отправитель выполняет следующие действия.

1. Дополняет сообщение $m$ нулями: $m||0^{k_1}$.
2. Генерирует случаю строку $r$ длины $k_0$ бит.
3. Вычисляет $X = m||0^{k_1} \oplus G(r)$.
4. Вычисляет $Y = r \oplus H(X)$.
5. Полагает $m' = X||Y$.
6. Выполняет зашифрование дополненного сообщения:

$$c' = (m')^e (mod\, n).$$

Для расшифрования шифртекста $c'$ получатель выполняет следующие действия.

1. Выполняет расшифрование дополненного сообщения:

$$m' = (c')^d (mod\, n) = X||Y.$$

2. Вычисляет значение $r = Y \oplus H(X)$.

3. Вычисляет значение $m||0^{k_1} = X \oplus G(r)$ и проверяет наличие $k_1$ нулей. Если условие не выполняется, то делает вывод о некорректности шифртекста. В противном случае, $m$ − результат расшифрования. ■

## Контрольные вопросы

1. Обеспечивают ли часто используемые открытые показатели вида $2^{16} + 1$ стойкость к методу криптоанализа на основе частично известных открытых текстов?

2. При какой длине дополнения по двум шифртекстам можно восстановить исходное сообщение при $e = 3$ и длине модуля 1024 бит?

3. Перечислите преимущества RSA-OAEP по сравнению с классической схемой RSA.

4. Какие существуют модификации RSA на основе схемы OAEP?

## Порядок выполнения работы

1. Разработать программу **П-1**, которая реализует атаку на криптосистему RSA для малого открытого показателя и короткого сообщения. Программа **П-1** должна принимать на вход составное число $n$, шифртексты $c_1$, $c_2$, открытый показатель $e$; на выходе – возвращать восстановленное сообщение.

2. Разработать программу **П-2**, которая реализует зашифрование и расшифрование файла согласно схеме преобразования RSA-OAEP. Зашифрование файла необходимо производить с использованием вспомогательного симметричного алгоритма AES-256 в режиме CBC, реализация симметричного алгоритма не требуется. Ключ шифрования симметричного алгоритма – случайный или изначально задан в программе. В режиме зашифрования программа должна принимать на вход файл сообщения (произвольного формата), открытый ключ RSA, ключ симметричного алгоритма; на выходе − возвращать зашифрованный файл, заголовок которого соответствует нотации ASN.1, описанной в *Приложении А*. В режиме расшифрования **П-2**

должна принимать на вход зашифрованный файл, закрытый ключ RSA; на выходе – возвращать расшифрованное сообщение.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) описание осуществленной атаки;

б) параметры схемы RSA, используемые при атаке;

в) использованные параметры для RSA-OAEP;

г) сгенерированный ключ для симметричного алгоритма;

д) содержание зашифрованного файла в шестнадцатеричном виде (при значительном размере файла – первые 80-100 байт);

е) содержимое расшифрованного файла.

3. Ответы на контрольные вопросы.
4. Выводы по работе.
5. Листинги программ.

*ЛАБОРАТОРНАЯ РАБОТА № 4*

# АТАКА ПО ВРЕМЕНИ НА КРИПТОСИСТЕМУ RSA

Цель работы – изучение атак на RSA, основанных на неправильной реализации криптосистемы, выполнение атаки по времени на криптосистему RSA.

## Теоретические сведения

Среди множества атак на криптосистемы существует класс атак по сторонним (побочным) каналам, учитывающих специфику практической реализации алгоритма. К ним, прежде всего, относятся атаки по времени исполнения, по энергопотреблению, электромагнитному излучению, на кэш, акустические атаки и др.

В целях оптимизации вычислений криптографический алгоритм может выполнять различное количество операций в зависимости от значения закрытого параметра и входных данных. Использование подобных особенностей позволяет извлекать задействованный при вычислениях закрытый ключ, например, путем анализа времени выполнения операций.

### 1. Модульное возведение в степень и метод Монтгомери

Процедура расшифрования в криптосистеме RSA основана на операции модульного возведения шифртекста в степень закрытого ключа. В целях увеличения производительности во многих версиях библиотек при расшифровании шифртекста $c$ вычисляются значения $m_1 \equiv c^{d_1} (mod\, p)$, $m_2 \equiv c^{d_2} (mod\, q)$, где $d_1, d_2$ – предвычисленные значения, полученные из закрытого ключа $d$, а затем исходное сообщение $m$ восстанавливается согласно китайской теореме об остатках.

Наиболее простая реализация алгоритма возведения числа в степень по модулю подразумевает последовательное возведение в квадрат и умножение в зависимости от значения бита показателя.

**Алгоритм 4.1.** Модульное возведение в степень

*Вход*. Элемент $c$, степень $d = (d_{k-1} \dots d_0)_2$, модуль $q$.

*Выход*. $r \equiv c^d (mod\, q)$.

1. Положить $r \leftarrow 1$.
2. Для $i = k-1, k-2, \dots, 0$ выполнить следующие действия.
   2.1. Вычислить $r \leftarrow r^2 (mod\, q)$.
   2.2. Если $d_i = 1$, то выполнить $r \leftarrow r \cdot c (mod\, q)$.
3. Результат: $r$. ■

Как видно из алгоритма 4.1, в зависимости от значения бита показателя выполняется только операция возведения в квадрат (если бит равен 0) или, в случае единичного бита, две операции: возведение в квадрат и умножение. После каждого шага результат приводится по модулю $q$.

Применение подобного алгоритма на практике для криптографических целей требует выполнения достаточно трудоемких операций (например, деления с остатком) над числами порядка 1024-4096 бит, что приводит к существенным временным и вычислительным затратам.

С целью оптимизации процедуры возведения в степень для умножения чисел и возведения в квадрат может быть использован метод Монтгомери, позволяющий обойтись без «дорогостоящей» операции деления с остатком. Данный метод рекомендуется применять, когда модульное умножение нужно производить многократно.

Пусть $N$ – модуль, элементы $T$ и $R$ такие, что $R = 2^w > N, \text{НОД}(N, R) = 1,\ 0 \le T < NR$. Так как числа $N$ и $R$ взаимно просты, можно найти такие $0 < R^{-1} < N, 0 < N' < R$, что $RR^{-1} - NN' = 1$. Произведение Монтгомери двух целых чисел $a$ и $b$ задается функцией $MP(a, b, N, R) \equiv abR^{-1} (mod\, N)$.

Для нахождения вычета $TR^{-1} (mod\, N)$ для произвольного значения $T$ используется преобразование Монтгомери, позволяющее ускорить вычисления путем замены операций по модулю $N$ на операции по модулю $R$, которые выполняются достаточно быстро, поскольку $R$ – степень двойки. Кроме того, подобное преобразование

можно использовать в произвольной системе счисления с основанием $B = 2^m$, тогда $R = B^n$.

**Алгоритм 4.2.** Преобразование Монтгомери

*Вход.* Элементы $T = (t_{2n-1}, \dots, t_0)_B, B = 2^m, R = B^n, N = (N_{n-1}, \dots, N_0)_B, N < R, N' \equiv -N^{-1} (mod\, B)$.

*Выход.* $TR^{-1} (mod\, N)$.

1. Для $i = 0,1, \dots, n-1$ выполнить следующие действия.
   1.1. Вычислить $m_i \leftarrow t_i \cdot N' (mod\, B)$.
   1.2. Положить $T \leftarrow T + m_i N B^i$.
2. Вычислить $t \leftarrow T/R$.
3. Если $t \geq N$, то $t \leftarrow t - N$.
4. Результат: $t$. ■

Тогда для вычисления произведения $a$ и $b$ по модулю $N$ достаточно перемножить их как целые числа, а затем выполнить преобразование Монтгомери для полученного значения.

Для произвольного числа $a < N$ можно определить специальное представление в виде $N$-вычета: $aR (mod\, N)$. Множество элементов $\{i \cdot R (mod\, N) | 0 \leq i \leq N-1\}$ является полной системой вычетов, т.е. содержит все числа от 0 до $N-1$. Таким образом, существует взаимно однозначное соответствие между кольцом $\mathbb{Z}/N\mathbb{Z}$ и системой $N$-вычетов. Используя данное свойство, можно определить процедуру умножения, действуя над элементами, заданными в форме $N$-вычетов. Тогда алгоритм умножения $N$-вычетов сводится к вычислению последовательности:

$$a \rightarrow aR (mod\, N) = MP(a, R^2, N, R),$$
$$b \rightarrow bR (mod\, N) = MP(b, R^2, N, R),$$
$$MP(aR, bR, N, R) = abR (mod\, N),$$
$$MP(abR, 1, N, R) = ab (mod\, N).$$

**Алгоритм 4.3.** Алгоритм умножения $N$-вычетов в системе счисления с основанием $B$

*Вход.* $aR (mod\, N) = (a_{n-1}, \dots, a_0)_B$, $bR (mod\, N) = (b_{n-1}, \dots, b_0)_B$, $B = 2^m$,$R = B^n$, $N' \equiv -N^{-1} (mod\, B)$.

*Выход.* $abR (mod\, N)$.

1. Положить $c \leftarrow 0, c = (c_{n-1}, \ldots, c_0)_B$.
2. Для $i = 0,1, \ldots, n-1$ выполнить следующие действия.
   2.1. Вычислить $u_i \leftarrow (c_0 + a_i b_0) N' (mod\, B)$.
   2.2. Вычислить $c \leftarrow (c + a_i (bR (mod\, N)) + u_i N)/B$.
3. Если $c \geq N$, то $c \leftarrow c - N$.
4. Результат: $c$. ■

На шаге 3 алгоритмов 4.2 и 4.3, если результат вычислений больше модуля $N$, выполняется дополнительная операция вычитания из результата значения модуля. Таким образом, в зависимости от значения $t$, время выполнения алгоритма может меняться.

Для криптографических приложений подобный временной эффект может быть использован для атак по побочным каналам, поэтому при реализации алгоритмов для умножения чисел рекомендуется использовать сбалансированный алгоритм Монтгомери без шага 3, ужесточив требования для границы: вместо $N < B^n$ рекомендуется использовать $N < B^{n-1}$.

**Пример 4.1.** Вычисление произведения чисел по модулю с помощью алгоритма Монтгомери

Пусть нужно найти произведение чисел $a = 468, b = 579$ по модулю $N = 751$, в качестве основания системы счисления возьмем $B = 2^5$. При реализации умножения методом Монтгомери, как правило, выбирают $N < B^n$, то есть в данном случае $n = 2$. Обозначим $R = B^n$. Тогда $N = (23,15)_B$, $N' \equiv -N^{-1} \equiv 17 (mod\, B)$.

Вычисляем целочисленное произведение $a$ и $b$:

$$T = (t_3, t_2, t_1, t_0)_B = ab = 270972 = (8,8,19,28)_B.$$

Реализация шага 1 алгоритма Монтгомери:

$$i = 0, m_0 = t_0 N' \equiv 28 (mod\, B),$$

$$T = T + m_0 N B^0 = (8,8,19,28)_B + 28 \cdot (23,15)_B \cdot B^0 = (8,29,5,0)_B;$$

$$i = 1, m_1 = t_1 N' \equiv 21 (mod\, B),$$

$$T = T + m_1 N B^1 = (8,29,5,0)_B + 21 \cdot (23,15)_B \cdot B^1 = (24,10,0,0)_B.$$

На шаге 2 выполняется деление на $R$ – сдвиг на два разряда вправо: $t = T/R = (24,10)_B = 778$. Поскольку получилось число, большее $N$, выполняется шаг 3:

$$t \leftarrow t - N = 778 - 751 = 27.$$

Таким образом, $TR^{-1} \equiv 27(mod\, N)$. Искомое произведение равно

$$(TR^{-1})R \equiv 27 \cdot 2^{10} \equiv 612(mod\, N).$$

В сбалансированной версии алгоритма Монтгомери необходимо увеличить число разрядов, выбрав $n$ из условия $N < B^{n-1}$. Тогда $n - 1 = 2$, а $R = B^n = B^3$.

В результате, $N = (0,23,15)_B$, и целочисленное произведение имеет длину $2n - 1 = 5$, $T = (t_4, t_3, t_2, t_1, t_0)_B = (0,8,8,19,28)_B$.

Из-за увеличения разрядности на шаге 1 алгоритма Монтгомери добавится еще одна итерация:

$$i = 0,\ T = T + m_0 N B^0 = (0,8,29,5,0)_B,$$
$$i = 1,\ T = T + m_1 N B^1 = (0,24,10,0,0)_B,$$
$$i = 2,\ m_2 = t_2 N' \equiv 10(mod\, B),$$
$$T = T + m_2 N B^2 = (0,24,10,0,0)_B + 10 \cdot (0,23,15)_B \cdot B^2 = (8,3,0,0,0)_B.$$

На шаге 2 получаем $t = T/R = (8,3)_B = 259 < N$, поэтому вычитание не требуется. Поскольку $R = B^3$, получаем $TR^{-1} \equiv 259(mod\, N)$, а искомое произведение совпадает с вычисленным ранее: $(TR^{-1})R \equiv 259 \cdot 2^{15} \equiv 612(mod\, N)$.■

## 2. Временная атака на криптосистему RSA

Согласно работе В. Шиндлера «A timing attack against RSA with the Chinese Remainder theorem», вероятность того, что при вычислении $g^d(mod\, q)$ алгоритмом Монтгомери понадобится операция дополнительного вычитания, равна $\frac{g\ (mod\, q)}{2R}$. Это значит, что при приближении $g$ к значению, кратному $q$, число вычитаний резко увеличивается, а при достижении этого значения − резко уменьшается. При $g < q$ расшифрование будет длиться медленнее, чем при $g > q$. Одна из атак по времени на криптосистему RSA, предложенная Д. Брамли и Д. Боне в работе «Remote timing attacks are practical», использует данный временной эффект для нахождения одного из множителей модуля $n = pq$, где $p < q$.

Начальное приближение $g$ для множителя $q$ полагается равным $2^{(log_2 n)/2}$. Далее строится предположение обо всех возможных ком-

бинациях старших битов (обычно 2–3), после чего измеряется время расшифрования.

Пусть $g$ – число, имеющее такие же $i$ старших битов, что и $q$; остальные биты установлены в 0.

**Алгоритм 4.4.** Атака по времени Брамли–Боне

*Вход*. Число $g = (q_0 q_1 \dots q_{i-1} 00\dots)_2$.

*Выход*. $q_i$.

1. Установить следующий бит числа $g$ равным 1 и обозначить $\overline{g} \leftarrow (q_0 q_1 \dots q_{i-1} 10\dots)_2$. Если у множителя $q$ бит $q_i$ равен 1, то $g < \overline{g} < q$; иначе $g < q < \overline{g}$.

2. Для $i = 0,1,2,\dots,l-1$ выполнить следующие действия.

   2.1. Вычислить $u_{g+i} = (g+i)R^{-1}\,(mod\, n)$, $u_{\overline{g}+i} = (\overline{g}+i)R^{-1}\,(mod\, n)$. Данный шаг необходим, так как при реализации RSA с умножением Монтгомери предварительно выполняется преобразование чисел $u_{g+i}, u_{\overline{g}+i}$ в форму Монтгомери: $u_{g+i}R = g+i, u_{\overline{g}+i}R = \overline{g}+i$.

   2.2. Измерить среднее время расшифрования для $u_{g+i}, u_{\overline{g}+i}$:

$$t_1^{(i)} = \frac{1}{s}\sum_{k=0}^{s-1} DecryptTime(u_{g+i}),$$

$$t_2^{(i)} = \frac{1}{s}\sum_{k=0}^{s-1} DecryptTime(u_{\overline{g}+i}).$$

3. Вычислить $T_g = \sum_{i=0}^{l} t_1^{(i)}$, $T_{\overline{g}} = \sum_{i=0}^{l} t_2^{(i)}$, $\Delta = |T_g - T_{\overline{g}}|$. Если $g < q < \overline{g}$, то величина $\Delta$ будет относительно «большой» и бит $q_i = 0$. Если $g < \overline{g} < q$, то величина $\Delta$ будет мала и бит $q_i = 1$.

4. Результат: $q_i$. ■

Для нахождения одного бита $q$ измеряется также время расшифрования для $l$ ближайших соседей: $g, g+1, g+2, \dots, g+l-1$, причем для каждого элемента $g+i$ выполняется $s$ запросов и вычисляется среднее время. Тогда суммарное число запросов для вычисления $T_g$ будет равно $s \cdot l$.

Таким образом, атака выполняет побитовое вскрытие множителя $q$ модуля $n$, начиная со старших разрядов. Для выполнения атаки нарушитель подает на вход алгоритму расшифрования текущее приближение $g, \overline{g}$ и измеряет время выполнения операции в зависимости

от входных данных. Если разница относительно большая, то устанавливается нулевой бит, иначе – единичный. Первые 2–3 бита необходимо подобрать.

После нахождения одного из множителей $q$, закрытый показатель $d$ можно найти путем вычисления второго множителя $p = n/q$ и мультипликативно обратного к открытому показателю $e$ по модулю $\varphi(n) = (p-1)(q-1)$: $d = e^{-1} (\mathrm{mod}\ \varphi(n))$.

## Контрольные вопросы

1. Сформулируйте методы противодействия атакам по времени для криптосистемы RSA.
2. Чем обусловлен выбор значения $R$ в алгоритме Монтгомери?

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, которая выполняет атаку по времени на некорректную реализацию RSA. Предоставленный бинарный файл выполняет расшифрование поданного на вход шифртекста алгоритмом RSA, используя фиксированный неизвестный закрытый ключ, и возвращает соответствующий открытый текст и время, затраченное на расшифрование.

В качестве входных данных программа **П-1** должна принимать параметры $l$ и $s$ (число запросов для вычисления одного бита), модуль $n$, на выходе – выдавать найденный множитель $q$, значение закрытого показателя $d$.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) найденное в результате временной атаки значение множителя $q$;

б) число запросов для вычисления одного бита числа $q$ (параметры $l$ и $s$);

в) пример вычислений трех-пяти битов числа $q$;

г) время, затраченное на реализацию атаки, и суммарное число запросов;

д) вычисленное значение закрытого показателя $d$.

3. Ответы на контрольные вопросы.
4. Выводы по работе.
5. Листинги программ.

*ЛАБОРАТОРНАЯ РАБОТА № 5*

# КРИПТОСИСТЕМЫ НА ОСНОВЕ ЗАДАЧИ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ

Цель работы – изучение протоколов шифрования с открытым ключом и электронной подписи, безопасность которых основана на задаче дискретного логарифмирования в конечном поле.

## Теоретические сведения

### 1. Протокол Эль-Гамаля шифрования с открытым ключом

Т. Эль-Гамаль предложил протокол шифрования с открытым ключом, основанный на задаче Диффи–Хеллмана: по данным образующей $a$ циклической группы $G$ и экспонентам $a^x$, $a^y$ с неизвестными показателями $x$, $y$ найти экспоненту $a^{xy}$. В протоколе Диффи–Хеллмана оба показателя $x$ и $y$ случайны. В протоколе Эль-Гамаля один показатель фиксирован и служит закрытым ключом расшифрования, а другой показатель является случайным. Рассмотрим обобщенный протокол шифрования с открытым ключом, который иногда называют «мета-протоколом» Эль-Гамаля.

Пусть $M$ – множество сообщений и $G$ – циклическая группа. Открытым ключом, общим для всех пользователей системы, являются группа $G$ и ее образующая $a$. Кроме того, существует общедоступная функция $f: M \times G \to G$, обратимая по первому аргументу (если известны значение функции и второй аргумент, то можно легко вычислить первый аргумент). Личным закрытым ключом является показатель $x$, личным открытым ключом – экспонента $b = a^x$.

**Протокол 5.1.** Обобщенный протокол Эль-Гамаля шифрования с открытым ключом

*Вход отправителя*. Группа $G$, функция $f: M \times G \to G$, образующая $a$, элемент $b = a^x$.

*Вход получателя*. Группа $G$, образующая $a$, логарифм $x$.

*Результат*. Зашифрование и расшифрование сообщения $m$.

Для зашифрования сообщения $m$ отправитель выполняет следующие действия.

1. Генерирует случайный показатель $y \in \mathbb{Z}$.
2. Вычисляет элементы $a^y, b^y \in G$ и значение $c \leftarrow f(m, b^y)$.

Шифртекстом является пара $(a^y, c)$.

Для расшифрования шифртекста $(a^y, c)$ получатель выполняет следующие действия.

1. Возводит $a^y$ в степень $x$, получает при этом $a^{xy} = b^y$.
2. Находит сообщение $m = f^{-1}(c, b^y)$.■

Вид функции $f$ зависит от группы $G$. Если $G = \mathbb{F}_p^*$ – мультипликативная группа простого поля, то в качестве $f$ могут использоваться функции $f(m, b^y) = mb^y \ (mod \ \ p)$, $f(m, b^y) = m + b^y \ (mod \ \ p)$, $f(m, b^y) = m - b^y \ (mod \ \ p)$ и т.п.

## 2. Протокол установления ключа Диффи-Хеллмана

Задача Диффи–Хеллмана полиномиально сводится к задаче дискретного логарифмирования: для заданных $a$, $b$ найти показатель $x$, для которого выполняется равенство $a^x = b$. Если по $a^x$ можно найти $x$, то по $a^y$ можно найти $a^{xy}$. Проблема обратной сводимости задачи дискретного логарифмирования к задаче Диффи–Хеллмана в общем случае пока не решена.

Задача Диффи-Хеллмана используется в протоколе установления сеансового ключа. Для того, чтобы установить общий ключ, участники протокола договариваются о несекретных параметрах: группе $G$ простого порядка $r$ и образующей $a$. Ключ устанавливается в ходе диалога.

**Протокол 5.2.** Протокол установления ключа Диффи-Хеллмана

*Вход участников A и B протокола*. Циклическая группа $G$ большого простого порядка $r$, образующая $a$ группы $G$: $G =< a >$.

*Результат*. Общий сеансовый ключ.

1. Участник $A$ вырабатывает случайный закрытый показатель $x$, вычисляет значение $a^x$ и посылает его участнику $B$.

2. Участник $B$ вырабатывает случайный закрытый показатель $y$, вычисляет значение $a^y$ и посылает его участнику $A$.

3. Участник $A$ возводит полученный от $B$ текст $a^y$ в степень $x$ и получает значение ключа $K_A = (a^y)^x$.

4. Участник $B$ возводит полученный от $A$ текст $a^x$ в степень $y$ и получает значение ключа $K_B = (a^x)^y$. ■

Из протокола 5.2 видно, что значение $K_A$, вычисляемое участником $A$, равно значению $K_B$, вычисляемое участником $B$, т.к. циклическая группа $G$, в которой выполняются операции, является абелевой.

В качестве $G$ может выступать любая группа, в которой задача дискретного логарифмирования является вычислительно сложной, а операция возведения в степень − достаточно эффективной, например, мультипликативная группа $\mathbb{F}_p^*$ простого поля, а также группа точек эллиптической кривой.

## 3. Бесключевое шифрование Месси–Омуры

Протокол Месси–Омуры требует диалога между отправителем и получателем и, следовательно, для его реализации необходим двунаправленный канал связи.

Отправитель и получатель договариваются о параметрах криптосистемы: циклической группе $G$ порядка $r$ и способе представления текста $m \in M$ в виде элемента группы $G$, т.е. о вычислимой в обе стороны функции $f: M \to G$.

**Протокол 5.3.** Бесключевое шифрование Месси–Омуры

*Вход отправителя и получателя*. Циклическая группа $G$ большого простого порядка $r$, вычислимая в обе стороны функция $f: M \to G$.

*Результат*. Зашифрование и расшифрование сообщения $m$.

1. Отправитель вычисляет $t \leftarrow f(m)$ для сообщения $m$. Если $t$ является единичным элементом $e$ группы $G$, то результат: сообщение зашифрованию не подлежит.

2. Отправитель генерирует случайный показатель $a \in \mathbb{F}_r^*$, вычисляет элемент $t^a$ группы $G$ и направляет элемент $t^a$ получателю.

3. Получатель вырабатывает случайный показатель $b$, обратимый по модулю $r$, вычисляет $(t^a)^b = t^{ab}$ и посылает элемент $t^{ab}$ отправителю.

4. Отправитель возводит полученный элемент в степень $a^{-1}(mod\, r)$: $(t^{ab})^{a^{-1}} = t^b$ и посылает $t^b$ получателю.

5. Получатель возводит полученный элемент в степень $b^{-1}(mod\, r)$: $(t^b)^{b^{-1}} = t$ и вычисляет $m \leftarrow f^{-1}(t)$. ■

В протоколе Месси–Омуры шифртекстами служат элементы $t^a$, $t^{ab}$, $t^b$. Протокол будет работать и тогда, когда порядок группы не является простым, но имеет большой простой делитель. Если порядок группы $G$ кроме большого простого делителя $r$ имеет один или несколько малых делителей $p_i$, то нарушитель может извлечь некоторую информацию об открытом тексте, вычислив для шифртекста характер степени $p_i$: $(t^a)^{\frac{\#G}{p_i}}$ (или $(t^b)^{\frac{\#G}{p_i}}$). Если характер для шифртекста равен 1, то и характер для открытого текста равен 1. Поэтому желательно использовать группу $G$ простого порядка.

Оригинальный протокол Месси–Омуры использует в качестве $G$ мультипликативную группу $\mathbb{F}_p^*$ простого поля; в роли функции $f$ может выступать тождественное отображение.

Этот протокол, очевидно, не обеспечивает аутентичность, так как в общем случае ни отправитель, не получатель не знают, с кем ведут общение. Кроме того, если нарушитель может включиться последовательно в канал связи, то он может осуществить дешифрование, выдавая себя за получателя и отключив от канала подлинного получателя. Иногда отсутствие аутентичности не является помехой, например, если нарушитель может только прослушивать канал связи, но не вмешиваться в его работу. Для обеспечения имитозащиты можно также применять электронную подпись.

Безопасность этого протокола основана на задаче Месси–Омуры: для заданных $t^a$, $t^{ab}$, $t^b$ найти $t$. Эта задача с полиномиальной сложностью сводится к задаче дискретного логарифмирования.

## 4. Протокол подписи Эль-Гамаля

Протокол подписи Эль-Гамаля в подгруппе простого порядка $r$ мультипликативной группы $\mathbb{F}_p^*$ простого поля строится следующим образом. Закрытым ключом подписи служит целое число $x$, $0 < x < r$, открытым ключом проверки подписи – простое число $p$, образующая $a$ подгруппы порядка $r$ и экспонента $b \equiv a^x \pmod{p}$.

**Протокол 5.4.** Схема подписи Эль-Гамаля

*Вход отправителя*. Закрытый ключ $x$.

*Вход получателя*. Характеристика поля $p$, образующая $a$ подгруппы простого порядка $r$, экспонента $b \equiv a^x \pmod{p}$.

*Результат*. Формирование и проверка подписи.

Для формирования подписи для сообщения $m$, $0 < m < r$, отправитель выполняет следующие действия.

1. Генерирует случайное число $k$, $1 < k < r$.
2. Полагает $w \leftarrow a^k \pmod{p}$.
3. Находит число $s$, решая сравнение $m \equiv xw + ks \pmod{r}$:

$$s \leftarrow (m - xw)k^{-1} \pmod{r}.$$

Подписью для сообщения $m$ является пара $(w, s)$.

Для проверки подписи получатель выполняет следующие действия.

1. Проверяет неравенство $w < p$. Если оно не выполнено, то результат: подпись недействительна.

2. Проверяет сравнение для экспонент: $a^m \equiv b^w w^s \pmod{p}$. Если оно выполняется, то результат: подпись подлинная, иначе результат: подпись недействительна. ■

Поскольку число $k$, используемое на этапе формирования подписи, взаимно просто с числом $r$, то $k$ обратимо по модулю $r$, т.е. число $s$ всегда существует и оно единственное.

При формировании подписи в протоколе Эль-Гамаля можно использовать предвычисления. Действительно, числа $k$ и $w$ никак не связаны с сообщением и могут быть вычислены заранее, что позволяет ускорить процедуру формирования подписи.

Протокол Эль-Гамаля можно модифицировать. Например, использовать в сравнении $m \equiv xw + ks \ (mod \ \ r)$ вместо сложения вычитание; менять местами $s$ и $m$; умножать слагаемые на ненулевые коэффициенты: $am \equiv bxw + cks \ (mod\, r)$; заменять линейное сравнение алгебраическим уравнением, имеющим единственное решение в поле вычетов по модулю $r$. В частности, при $r \equiv 5 \ (mod\ 6)$ любое сравнение третьей степени всегда имеет единственное решение в группе $\mathbb{F}_r^*$, поэтому можно использовать сравнение $m^3 \equiv xw^3 + ks^3 \ (mod\, r)$, тогда проверка подписи сведется к проверке сравнения $a^{m^3} \equiv b^{w^3} + w^{s^3} \ (mod\, p)$. Можно возводить в степень не все коэффициенты, а также использовать произвольную нечетную степень $t$ такую, что $\text{НОД}(r-1, t) = 1$, так как отображение $x \to x^t \ (mod\, r)$ является взаимно однозначным и вычислимым в обе стороны.

## Контрольные вопросы

1. Пусть $G$ – конечная циклическая группа с заданной образующей. Порядок группы $G$ известен и делится на простое число $r$. Как найти образующую циклической подгруппы порядка $r$?
2. Покажите, что задача Месси–Омуры сводится к задаче дискретного логарифмирования.
3. Подвержен ли протокол шифрования Эль-Гамаля к атаке на основе подобранных шифртекстов?
4. Перечислите задачи, положенные в основу схемы подписи Эль-Гамаля. Почему важно, чтобы период генератора случайных чисел, используемого в схеме подписи, был достаточно большим?

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, которая, в соответствии с вариантом, реализует одну из криптосистем на основе задачи дискретного логарифмирования.

1. Протокол Эль-Гамаля шифрования с открытым ключом. Программа, по крайней мере, должна содержать три функции: генерации параметров криптосистемы, зашифрования и расшифрования

файла. Программа должна допускать возможность использования различных ключей.

2. Протокол установления ключа Диффи–Хеллмана. Протокол требует диалога между взаимодействующими сторонами, поэтому передача данных должна производиться посредством сетевого взаимодействия сторон. Необходимо использовать протокол TCP; хост и порт сервера должны задаваться в конфигурации клиента. TCP-соединение, открытое в начале взаимодействия, используется клиентом и сервером до окончания взаимодействия (механизм «keep-alive»).

3. Протокол бесключевого шифрования Месси–Омуры. Протокол требует диалога между взаимодействующими сторонами, поэтому передача данных должна производиться посредством сетевого взаимодействия сторон. Необходимо использовать протокол TCP, хост и порт сервера должны задаваться в конфигурации клиента. TCP-соединение, открытое в начале взаимодействия, используется клиентом и сервером до окончания взаимодействия (механизм «keep-alive»).

4. Схема подписи Эль-Гамаля. Программа, по крайней мере, должна содержать три функции: генерации параметров криптосистемы, формирования и проверки подписи. Программа также должна допускать возможность использования различных ключей. При подписи файла используется алгоритм хэширования SHA-256, причем результат хэширования файла участвует в алгоритме подписи как $m = h(\text{file})(mod\ r)$, где $h()$ – хэш-функция, $m$ – сообщение, $r$ – порядок мультипликативной подгруппы простого поля.

Описание ASN.1-структуры для каждой криптосистемы представлено в *Приложении В*.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) сравнительный анализ протоколов RSA и протокола Эль-Гамаля (варианты 1–3, 8–10, согласно *Приложению В*). Критерии сравнения: простота реализации, производительность;

б) полученные параметры криптосистемы;

в) сгенерированный (либо установленный) ключ симметричного алгоритма (кроме вариантов 8–10). Для схемы Эль-Гамаля – хэш-образ сообщения (варианты 8–10);

г) содержимое исходного открытого файла. При значительном размере файла – первые 80–100 байт.

д) содержимое зашифрованного файла в шестнадцатеричном виде (при значительном размере файла – первые 80–100 байт) (варианты 1–3). Полные журналы взаимодействия сторон в шестнадцатеричном виде (варианты 4–7). Содержимое файла с подписью (варианты 8–10).

3. Ответы на контрольные вопросы.

4. Выводы по работе, где сделать предположения о возможных уязвимостях разработанного программного обеспечения и о путях их устранения. Требования к параметрам реализованного протокола, выполнение которых может обеспечить максимальную стойкость к известным методам логарифмирования.

5. Листинг программы.

*ЛАБОРАТОРНАЯ РАБОТА № 6*

# ПРОТОКОЛ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ ГОСТ Р 34.10–2018

Цель работы – изучение протокола электронной цифровой подписи ГОСТ Р 34.10–2018, безопасность которого основана на задаче дискретного логарифмирования в группе точек эллиптической кривой.

## Теоретические сведения

Эллиптические кривые над конечными полями в настоящее время представляют наиболее популярные структуры для построения криптографических алгоритмов. Это объясняется тем, что эллиптические кривые позволяют реализовать весь спектр криптографических протоколов с открытым ключом; задача дискретного логарифмирования на эллиптической кривой имеет экспоненциальную стойкость, которая практически не снижается во времени.

Действующий стандарт электронной цифровой подписи (ЭЦП) ГОСТ Р 34.10–2018, как и стандарт подписи США DSS, использует эллиптические кривые над конечными полями.

### 1. Эллиптические кривые в форме Вейерштрасса

Пусть $E(K)$ – эллиптическая кривая над полем $K$, заданная уравнением в форме Вейерштрасса

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

где частные производные по $x$ и по $y$ полинома, задающего кривую, не обращаются в нуль одновременно ни в одной точке кривой даже при переходе к алгебраически замкнутому полю.

В криптографических алгоритмах на эллиптических кривых в качестве поля $K$ чаще всего используется конечное поле $\mathbb{F}_p$. Если характеристика $p$ отлична от 2 и 3, уравнение кривой можно записать в виде

$$y^2 = x^3 + Ax + B, \quad (1)$$

где $4A^3 + 27B^2 \neq 0\ (mod\ p)$.

Для эллиптической кривой $E(\mathbb{F}_p)$ определен инвариант

$$j = 12^3 \cdot 4A^3(4A^3 + 27B^2)^{-1}(mod\ p).$$

Точки эллиптической кривой образуют абелеву группу с геометрическим законом сложения, нулем группы является бесконечно удаленная точка $P_\infty$, точки $(x, y)$ и $(x, -y)$ являются противоположными.

Для сложения точек $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ через них проводят секущую, которая пересекает кривую в третьей точке $P_3$ с координатами $(x_3, -y_3)$. Тогда точка $(x_3, y_3)$, противоположная к точке $P_3$, по определению является суммой точек $P_1$ и $P_2$. Закон сложения описывается следующими формулами. Если $P_1 = P_2$, то координаты точки $P_3 = P_1 + P_2 = 2P_1$ равны

$$x_3 = \lambda^2 - 2x_1(mod\ p),$$
$$y_3 = \lambda(x_1 - x_3) - y_1(mod\ p),$$

где угловой коэффициент равен $\lambda = (3{x_1}^2 + A)(2y_1)^{-1}(mod\ p)$.

Если $P_1 \neq \pm P_2$, то координаты точки $P_3 = P_1 + P_2$ равны

$$x_3 = \lambda^2 - x_1 - x_2(mod\ p),$$
$$y_3 = \lambda(x_1 - x_3) - y_1(mod\ p),$$

где угловой коэффициент равен $\lambda = (y_2 - y_1)(x_2 - x_1)^{-1}(mod\ p)$.

Формулы удвоения и сложения точек содержат операцию модульного обращения, которая является достаточно трудоемкой (выполняется с использованием расширенного алгоритма Евклида). Поэтому зачастую эллиптическую кривую записывают в проективной форме. Переход к проективной плоскости осуществляется заменой переменных: $x = XZ^{-1}, y = YZ^{-1}$.

Кривая (1) в нормальной форме Вейерштрасса в проективной плоскости задается уравнением

$$y^2Z = X^3 + AXZ^2 + BZ^3\ (mod\ \ p) \quad (2)$$

с отношением эквивалентности

$$(X, Y, Z) \sim (uX, uY, uZ),$$

где $u \in \mathbb{F}_p^*$.

Любой класс эквивалентности троек $(X, Y, Z) \sim (uX, uY, uZ)$ с ненулевой $Z$-координатой содержит единственную тройку вида $(X, Y, 1)$. Переход от точки кривой в аффинной плоскости ( к точке кривой в проективной плоскости ( производится при помощи добавления к координатам третьей координаты $Z = 1$.

$$(\boldsymbol{x}, \boldsymbol{y}) \to (\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{1}).$$

Обратный переход от ( к ( осуществляется с помощью «сокращения» координат

$$(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) = \left(\frac{\boldsymbol{X}}{\boldsymbol{Z}}, \frac{\boldsymbol{Y}}{\boldsymbol{Z}}, \boldsymbol{1}\right) \to (\boldsymbol{x} = \boldsymbol{X}, \boldsymbol{y} = \boldsymbol{Y}).$$

Для каждой точки $P$ определена противоположная точка, проективные координаты которой удовлетворяют соотношению

$$-\boldsymbol{P} = -(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) = (\boldsymbol{X}, -\boldsymbol{Y}, \boldsymbol{Z}).$$

Для кривой ( формулы удвоения в проективных координатах $(X_3, Y_3, Z_3) = 2(X_1, Y_1, Z_1)$ имеют вид:

$$X_3 \equiv 2Y_1Z_1((3{X_1}^2 + A{Z_1}^2)^2 - 8X_1{Y_1}^2Z_1) \ (mod\ p),$$
$$Y_3 \equiv 4{Y_1}^2Z_1(3X_1(3{X_1}^2 + A{Z_1}^2) - 2{Y_1}^2Z_1) - (3{X_1}^2 + A{Z_1}^2)^3 (mod\ p),$$
$$Z_3 \equiv 8{Y_1}^3{Z_1}^3 (mod\ p).$$

Формулы сложения двух различных точек в проективных координатах $(X_3, Y_3, Z_3) = (X_1, Y_1, Z_1) + (X_2, Y_2, Z_2)$ для этой же кривой имеют вид:

$$X_3 \equiv (X_2Z_1 - X_1Z_2)(Z_1Z_2(Y_2Z_1 - Y_1Z_2)^2 -$$
$$-(X_2Z_1 + X_1Z_2)(X_2Z_1 - X_1Z_2)^2) \ (mod\ p),$$
$$Y_3 \equiv (X_2Z_1 - X_1Z_2)^2\big(Y_2Z_1(X_2Z_1 + 2X_1Z_2) - Y_1Z_2(X_1Z_2 + 2X_2Z_1)\big) -$$
$$-Z_1Z_2(Y_2Z_1 - Y_1Z_2)^3 (mod\ p),$$
$$Z_3 \equiv Z_1Z_2(X_2Z_1 - X_1Z_2)^3 \ (mod\ p).$$

## 2. Умножение точки на число

Операция умножения точки $P$ эллиптической кривой на целое число $k$ представляет собой $k$-кратное применение групповой операции к точке $P$: $kP = \underbrace{P + P + \ldots + P}_{k \text{ раз}}$, при этом $-kP = -(kP) = k(-P)$.

Наиболее простым методом умножения точки на число является бинарное умножение. При использовании бинарного умножения чис-

ло, на которое умножается точка, представляется в двоичной системе счисления.

**Алгоритм 6.1.** Бинарное умножение точки на число

*Вход:* Точка эллиптической кривой $P$. Целое положительное число $k$.

*Выход:* Произведение $kP$.

1. Если $k = 0$, то результат: $P_\infty$ – бесконечно удаленная точка.
2. Представить число $k$ в двоичной системе счисления: $k = (k_{t-1} \dots k_1 k_0)_2$.
3. Положить $Q \leftarrow P_\infty$.
4. Для $i = t-1, \dots, 0$ выполнить:

4.1. $Q \leftarrow 2Q$ (удвоение точки).

4.2. Если $k_i \neq 0$, положить $Q \leftarrow Q + P$ (сложение точек).

5. Результат $Q$. ■

Алгоритм бинарного умножения точки на число во многом схож с алгоритмом модульного возведения в степень.

## 3. Стандарт подписи ГОСТ Р 34.10–2018

Стандарт ГОСТ Р 34.10–2018 подготовлен на основе стандарта ГОСТ Р 34.10–2012. Криптографическая стойкость схемы цифровой подписи определяется сложностью решения задачи дискретного логарифмирования в группе точек эллиптической кривой, а также стойкостью используемой в протоколе хэш-функции ГОСТ Р 34.11–2018.

Задача дискретного логарифмирования в группе точек эллиптической кривой формулируется следующим образом: для известной точки $P \in E(\mathbb{F}_p)$ (образующей подгруппы группы точек эллиптической кривой $E$) и данной точки $Q \in \langle P \rangle$ найти показатель $d$ такой, что $Q = dP$.

Протокол подписи стандарта ГОСТ Р 34.10–2018 предусматривает использование эллиптической кривой $E$ в форме (1) над простым полем $\mathbb{F}_p$ $(p > 3)$. Эллиптическая кривая задается инвариантом или коэффициентами $A, B$, вычисление которых возможно по известному значению инварианта:

$$\begin{cases} A = 3k \ (mod\ p), \\ B = 2k \ (mod\ p), \end{cases}$$

где $k = j(1728 - j)^{-1}(mod\ p), j \neq 0,\ j \neq 1728.$

Параметры цифровой подписи:

1) $p$ – простое число (модуль эллиптической кривой);

2) эллиптическая кривая $E$ (коэффициенты $A, B$ или $j(E)$);

3) $m$ – порядок группы точек эллиптической кривой $\#E(\mathbb{F}_p)$;

4) $q$ – простой делитель порядка группы точек $m$: $m = n \cdot q$, $n \in \mathbb{Z}, n > 1$. Длина числа $q$ определяется в соответствии со стандартом на хэш-функцию ГОСТ Р 34.11–2018: $2^{254} < q < 2^{256}$ или $2^{508} < q < 2^{512}$;

5) точка $P \in E(\mathbb{F}_p)$, $P \neq P_\infty$, $qP = P_\infty$, $\#\langle P \rangle = q$.

Дополнительные требования:

1) $p^t \neq 1 \ (mod\ q), \forall t = 1, \dots, D$, где $D = 31$, если $2^{254} < q < 2^{256}$; $D = 131$, если $2^{508} < q < 2^{512}$;

2) $m \neq p$;

3) $j(E) \neq 0,\ 1728.$

В основу протокола электронной цифровой подписи ГОСТ Р 34.10–2018 положен протокол Эль-Гамаля. Пусть $M$ – подписываемое сообщение, $h$ – хэш-функция по ГОСТ Р 34.11–2018, $\{E(\mathbb{F}_p), P, q\}$ – параметры цифровой подписи; число $d$ – ключ подписи $(0 < d < q)$, $Q = dP$ – ключ проверки.

**Протокол 6.1.** Схема подписи ГОСТ Р 34.10–2018

*Вход отправителя*. Закрытый ключ $d$, параметры подписи, сообщение $M$.

*Вход получателя*. Открытый ключ $Q$, параметры подписи.

*Результат*. Формирование и проверка подписи для сообщения $M$.

Для формирования подписи для сообщения $M$ отправитель выполняет следующие действия.

1. Вычисляет хэш-образ сообщения $M$: $\bar{h} \leftarrow h(M)$.

2. Вычисляет число $\alpha \in \mathbb{Z}$, двоичным представлением которого является вектор $\bar{h}$. Находит $e \equiv \alpha(mod\ q)$, при $e = 0$ полагает $e \leftarrow 1$.

3. Вырабатывает случайный показатель $k \in \mathbb{Z}$, $0 < k < q$.

4. Вычисляет точку $C \leftarrow kP$, $C = (x_C, y_C)$, $r \equiv x_C (mod\ q)$. При $r = 0$ возвращается на шаг 3.

5. Вычисляет $s \leftarrow (rd + ke)\ (mod\ q)$, при $s = 0$ возвращается на шаг 3.

6. Вычисляет двоичные векторы, соответствующие числам $r, s$: $r \to \overline{r}, s \to \overline{s}$. Цифровая подпись имеет вид $\xi = (\overline{r}||\overline{s})$.

Для проверки подписи получатель выполняет следующие действия.

1. Переводит двоичные векторы в числа: $\overline{r} \to r, \overline{s} \to s$, где $r, s \in \mathbb{Z}$. Проверяет выполнение неравенств $0 < r < q$, $0 < s < q$. Если хотя бы одно из них не выполняется, то подпись неверна.

2. Вычисляет хэш-образ сообщения: $\bar{h} \leftarrow h(M)$.

3. Вычисляет число $\alpha \in \mathbb{Z}$, двоичным представлением которого является вектор $\bar{h}$. Находит $e \equiv \alpha (mod\ q)$, при $e = 0$ полагает $e \leftarrow 1$.

4. Вычисляет $v \equiv e^{-1} (mod\ q)$.

5. Вычисляет $z_1 \equiv sv (mod\ q)$, $z_2 \equiv -rv (mod\ q)$.

6. Находит точку $C = z_1 P + z_2 Q$, $C = (x_C, y_C)$. $R \equiv x_C (mod\ q)$.

7. Проверяет выполнение равенства $R \stackrel{?}{=} r$. Если оно выполнено, то подпись принимается, в противном случае – подпись неверна. ■

## Контрольные вопросы

1. Перечислите преимущества криптосистем на эллиптических кривых по сравнению с другими криптосистемами.

2. Почему в стандарте ГОСТ Р 34.10–2018 введено требование $\#E(\mathbb{F}_p) \neq p$?

3. Если нарушитель имеет возможность обращать хэш-функцию, как он может подделать сообщение и подпись?

4. Почему случайный показатель $k$ не должен повторяться в течение времени жизни ключа?

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, реализующую протокол подписи согласно ГОСТ Р 34.10–2018. Описание формата файла подписи представлено в *Приложении Г*. Параметры криптосистемы соответствуют варианту задания и приведены в *Приложении Д*. Программа должна поддерживать функции формирования и проверки подписи, а также допускать возможность использования различных ключей.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) использованные параметры криптосистемы;

б) пример файла, используемого для подписи. При значительном размере файла – первые 80–100 байт;

в) хэш-образ сообщения (файла);

г) содержимое файла подписи в шестнадцатеричном виде (при значительном размере файла – первые 80–100 байт).

3. Ответы на контрольные вопросы.
4. Выводы по работе, где сделать предположения о возможных уязвимостях разработанного программного обеспечения и о путях их устранения.
5. Листинг программы.

*ЛАБОРАТОРНАЯ РАБОТА № 7*

# РЕШЕНИЕ ЗАДАЧИ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ НА ЭЛЛИПТИЧЕСКОЙ КРИВОЙ

Цель работы – изучение методов дискретного логарифмирования на эллиптической кривой, реализация методов Полларда и Полига–Хеллмана в группе точек эллиптической кривой.

## Теоретические сведения

Задача дискретного логарифмирования в группе точек эллиптической кривой может решаться универсальными методами: Полига–Хеллмана, встречи посередине, «giant step – baby step», встречи на случайном дереве и методом Полларда, – предложенными первоначально для логарифмирования в мультипликативной группе простого поля.

### 1. Метод Полларда

Пусть дана кривая $E(K)$ над конечным полем $K$, образующая точка $P$, $\#\langle P\rangle = q$, точка $Q \in E(K)$, $Q = dP$. Основная цель метода Полларда состоит в поиске таких пар $(\alpha', \beta')$ и $(\alpha'', \beta'')$, что выполняется равенство

$$\alpha' P + \beta' Q = \alpha'' P + \beta'' Q.$$

Тогда $(\alpha' - \alpha'')P = (\beta'' - \beta')Q = (\beta'' - \beta')dP$, и логарифм $d$ может быть вычислен как

$$d \equiv (\alpha' - \alpha'')(\beta'' - \beta')^{-1} (mod\ q).$$

В методе Полларда используется ветвящееся псевдослучайное отображение, обладающее сжимающими свойствами и вычислимостью логарифма.

Для организации ветвления на эллиптических кривых можно использовать следующее разбиение: $\langle P\rangle = S_1 \cup S_2 \cup \ldots \cup S_L$, где $L = 16$ или $L = 32$. Например, если $L = 32$, то будем считать, что точка

$T \in \langle P \rangle$ принадлежит множеству $S_j$ при $x_T \equiv j(mod\ 32)$. Значение функции ветвления $H: \langle P \rangle \to \{1, \ldots, L\}$ будет равно $H(T) = j$.

Пусть $a_j, b_j \in \mathbb{Z}/q\mathbb{Z}$, $j = \{1, \ldots, 32\}$. Тогда отображение $f: \langle P \rangle \to \langle P \rangle$ определяется следующим образом

$$f(T) = T + a_j P + b_j Q, j = H(T).$$

Очевидно, что свойство вычислимости логарифма при этом сохраняется: пусть $T = \alpha P + \beta Q$, тогда

$$f(T) = T + a_j P + b_j Q = (\alpha + a_j)P + (\beta + b_j)Q = \alpha' P + \beta' Q.$$

**Алгоритм 7.1.** Логарифмирование на эллиптической кривой методом Полларда

*Вход*. Кривая $E(K)$ над конечным полем $K$, образующая точка $P$, точка $Q \in E(K)$, простой порядок $q$ группы $\langle P \rangle$.

*Выход*. Логарифм $d$ такой, что $P = dQ$.

1. Выбрать $L$.
2. Выбрать функцию $H: \langle P \rangle \to \{1, \ldots, L\}$.
3. Для $j = 1, \ldots, L$ выполнить следующие действия.

3.1. Случайным образом выбрать $a_j, b_j \in \mathbb{Z}/q\mathbb{Z}$.

3.2. Вычислить $R_j = a_j P + b_j Q$.

4. Случайным образом выбрать $\alpha', \beta' \in \mathbb{Z}/q\mathbb{Z}$ и вычислить точку $T' \leftarrow \alpha' P + \beta' Q$. Положить $T'' \leftarrow T', \alpha'' \leftarrow \alpha', \beta'' \leftarrow \beta'$.
5. Повторять следующие действия, пока не получим $T' = T''$.

5.1. Положить $j \leftarrow H(T'), T' \leftarrow T' + R_j$,

$\alpha' \leftarrow \alpha' + a_j(mod\ q), \beta' \leftarrow \beta' + b_j(mod\ q)$.

5.2. Положить $j \leftarrow H(T''), T'' \leftarrow T'' + R_j, \alpha'' \leftarrow \alpha'' + a_j(mod\ q), \beta'' \leftarrow \beta'' + b_j(mod\ q), j \leftarrow H(T''), T'' \leftarrow T'' + R_j, \alpha'' \leftarrow \alpha'' + a_j(mod\ q), \beta'' \leftarrow \beta'' + b_j(mod\ q)$.

6. Если $\alpha' = \alpha'', \beta' = \beta''$, то перейти на шаг 1. Иначе положить

$$d \equiv (\alpha' - \alpha')(\beta'' - \beta')^{-1}(mod\ q).$$

7. Результат: $d$.■

**Пример 7.1.** Логарифмирование на эллиптической кривой методом Полларда

Пусть $E(\mathbb{F}_{307})$: $y^2 = x^3 + x + 11, P = (306,304)$, $q = \#\langle P\rangle = 167$, $Q = (146,65)$. Нужно найти логарифм $d$ такой, что $Q = dP$.

Выбираем функцию $H: \langle P\rangle \to \{1,2,3,4\}, L = 4$. Генерируем $a_j, b_j \in \mathbb{Z}/q\mathbb{Z}$ и вычисляем точки $R_j = a_jP + b_jQ$, $j = 1,\dots,L$. Результаты предвычислений приведены в таблице 7.1.

Случайным образом выбираем значения $\alpha' = 152, \beta' = 113$, тогда точка $T' = (19,83)$. Строим цепочки отображений $f(T')$ и $f(f(T'))$ до получения равенства. Результаты вычислений приведены в таблице 7.2.

Таблица 7.1 – Результаты предвычислений

| $j$ | $a_j$ | $b_j$ | $R_j$ |
|---|---|---|---|
| 1 | 71 | 62 | (175, 252) |
| 2 | 71 | 157 | (275, 260) |
| 3 | 93 | 50 | (275, 260) |
| 4 | 156 | 76 | (245, 245) |

Таблица 7.2 – Решение задачи дискретного логарифмирования на эллиптической кривой методом Полларда

| $i$ | $\alpha'$ | $\beta'$ | $T'$ | $\alpha''$ | $\beta''$ | $T''$ |
|---|---|---|---|---|---|---|
| 1 | 141 | 22 | (47, 280) | 130 | 98 | (292, 199) |
| 2 | 130 | 98 | (292, 199) | 23 | 69 | (275, 47) |
| 3 | 34 | 160 | (171, 208) | 83 | 135 | (245, 245) |
| 4 | 23 | 69 | (275, 47) | 143 | 34 | (5, 270) |
| 5 | 12 | 145 | (113, 296) | 36 | 100 | (47, 280) |
| 6 | 83 | 135 | (245, 245) | 96 | 71 | (171, 208) |
| 7 | 154 | 125 | (171, 99) | 74 | 56 | (113, 296) |
| 8 | 143 | 34 | (5, 270) | 49 | 36 | (171, 99) |
| **9** | **47** | **24** | **(19, 83)** | **109** | **102** | **(19, 83)** |

Встреча $T' = T''$ произошла на шаге 9 (в таблице обозначено полужирным шрифтом). Таким образом, $47P + 24Q = 109P + 102Q$.

Откуда находим $d = (47 - 109)(102 - 24)^{-1}(mod\ 167) = 72(\mathrm{m}od\ 167)$. ■

## 2. Метод Полига–Хеллмана

Пусть дана точка $P \in E(K)$, порядок $q$ циклической группы $\langle P \rangle$ раскладывается на множители $q = \prod_{j=1}^{t} q_j^{\alpha_j}$. Суть метода Полига–Хеллмана заключается в вычислении логарифмов по модулю $q_j^{\alpha_j}$: $d_j \equiv d\ (mod\, q_j^{\alpha_j}), j = 1, \dots, t$. Тогда искомый логарифм $d = log_P Q$ можно восстановить с помощью китайской теоремы об остатках.

Каждое число $d_j$ можно представить в виде

$$d_j = z_0 + z_1 q_j + z_2 q_j^2 + \dots + z_{\alpha_j - 1} q_j^{\alpha_j - 1}, z_i \in [0, q_j - 1].$$

Зададим начальную точку $P_0 = \frac{q}{q_j} P$ и обозначим $Q_0 = \frac{q}{q_j} Q$. Так как $\#\langle P_0 \rangle = q_j$, то

$$Q_0 = \frac{q}{q_j} Q = d_j \left(\frac{q}{q_j} P\right) = d_j P_0 = (z_0 + z_1 q_j + z_2 q_j^2 + \dots) P_0 =$$
$$= z_0 P_0 + z_1 \underbrace{q_j P_0}_{P_\infty} + z_2 \underbrace{q_j^2 P_0}_{P_\infty} + \dots = z_0 P_0.$$

Решая задачу дискретного логарифмирования в группе $\langle P_0 \rangle$ малого порядка (например, методом Полларда), вычисляем значение $z_0 = log_{P_0} Q_0$. Далее

$$Q_1 = \frac{q}{q_j^2} \left( \underbrace{Q}_{d_j P} - z_0 P \right) = (d_j - z_0) \left(\frac{q}{q_j^2} P\right) = z_1 \left(\frac{q}{q_j} P\right) + z_2 \underbrace{(qP)}_{P_\infty} + \dots = z_1 P_0.$$

Тогда $z_1 = log_{P_0} Q_1$. На каждой итерации находим $z_k = log_{P_0} Q_k$, вычисляя точку

$$Q_k = \frac{q}{q_j^{k+1}} (Q - z_0 P - z_1 q_j P - \dots - z_{k-1} q_j^{k-1} P), k = 1, \dots, \alpha_j - 1.$$

Данная процедура повторяется для $j = 1, \dots, t$. В итоге получается система сравнений:

$$\begin{cases} d \equiv d_1 (mod\ q_1^{\alpha_1}), \\ d \equiv d_2 (mod\ q_2^{\alpha_2}), \\ \dots \\ d \equiv d_t (mod\ q_t^{\alpha_t}). \end{cases}$$

Применяя китайскую теорему об остатках, восстанавливаем значение $d$. Таким образом, задача дискретного логарифмирования сводится к $t$ частным задачам дискретного логарифмирования меньшей размерности.

**Алгоритм 7.2.** Логарифмирование на эллиптической кривой методом Полига–Хеллмана

*Вход*. Кривая $E(K)$ над конечным полем $K$, образующая точка $P$, точка $Q \in E(K)$, разложение порядка группы $\#\langle P \rangle = q = \prod_{j=1}^{t} q_j^{\alpha_j}$.

*Выход*. Логарифм $d$ такой, что $P = dQ$.

1. Для $j = 1, 2, \dots, t$ выполнить следующие действия.
   1.1. Положить $p \leftarrow q_j, a \leftarrow \alpha_j$.
   1.2. Положить $S \leftarrow P_\infty, z_{-1} \leftarrow 0$.
   1.3. Вычислить $P_0 \leftarrow \frac{q}{p} P$.
   1.4. Для $k = 0, 1, \dots, a - 1$ выполнить следующие действия.
      1.4.1. Положить $S \leftarrow S + z_{k-1} p^{k-1} P$, $Q_k \leftarrow \frac{q}{p^{k+1}} (Q - S)$.
      1.4.2. Вычислить логарифм $z_k \leftarrow log_{P_0} Q_k$ любым алгоритмом (например, алгоритмом Полларда).
   1.5. Положить $d_j \leftarrow z_0 + z_1 p + \dots + z_{a-1} p^{a-1} (mod\ p^a)$.
2. Восстановить логарифм $d$ из $d_j$ по китайской теореме об остатках.
3. Результат: $d$. ■

**Пример 7.2.** Логарифмирование на эллиптической кривой методом Полига–Хеллмана

Пусть эллиптическая кривая $E(\mathbb{F}_{9103})$ задана уравнением $y^2 = x^3 + 3x + 16$. Пусть точка $P = (4515,5197) \in E(\mathbb{F}_{9103})$, ее порядок равен $q = \#\langle P \rangle = 2^2 \cdot 3^2 \cdot 11 \cdot 23$.

Пусть $Q = (8132, 7165)$. Найдем логарифм $d$ такой, что $P = dQ$.

Для вычисления $d_1 \equiv d(mod\ 2^2)$ запишем $d_1$ в виде $d_1 = z_0 + z_1 \cdot 2$. Вычисляем точки:

$$P_0 = \frac{q}{q_j} P = 2 \cdot 3^2 \cdot 11 \cdot 23P = (1814,0),$$

$$Q_0 = \frac{q}{q_j} Q = 2 \cdot 3^2 \cdot 11 \cdot 23Q = (1814,0).$$

Находим $z_0 = log_{P_0} Q_0 = 1$. Продолжая вычисление

$$Q_1 = 3^2 \cdot 11 \cdot 23(Q - z_0 P) = P_\infty,$$

получаем $z_1 = log_{P_0} Q_1 = 0$. Тогда $d_1 = 1 + 0 \cdot 2 = 1$.

Далее вычислим значение $d_2 \equiv d(mod\ 3^2)$, где $d_2 = z_0 + z_1 \cdot 3$. Задаем начальные точки $P_0 = 2^2 \cdot 3 \cdot 11 \cdot 23P = (4568,6980)$, $Q_0 = 2^2 \cdot 3 \cdot 11 \cdot 23Q = (4568,2123)$, тогда $z_0 = 2$. Продолжая вычисление

$$Q_1 = 2^2 \cdot 11 \cdot 23(Q - z_0 P) = (4568,6980)$$

получаем $z_1 = log_{P_0} Q_1 = 1$. Тогда $d_2 = 2 + 1 \cdot 3 = 5$.

Аналогично вычисляется $d_3 \equiv d(mod\ 11)$:$P_0 = 2^2 \cdot 3^2 \cdot 23P = (4360,7730)$,

$$Q_0 = 2^2 \cdot 3^2 \cdot 23P = (2469,6425),$$

$$z_0 = log_{P_0} Q_0 = 6, d_3 = z_0 = 6.$$

И для случая $d_4 \equiv d(mod\ 23)$:$P_0 = 2^2 \cdot 3^2 \cdot 11P = (6794,3772)$,

$$Q_0 = 2^2 \cdot 3^2 \cdot 11Q = (2343,6870),$$

$$z_0 = log_{P_0} Q_0 = 3, d_4 = z_0 = 3.$$

Получаем систему сравнений: $d \equiv 1(mod\ 2^2), d \equiv 5(mod\ 3^2), d \equiv 6(mod\ 11), d \equiv 3(mod\ 23)$. Восстанавливаем по китайской теореме об остатках значение логарифма: $d \equiv 1337(mod\ 9108)$. ■

## Контрольные вопросы

1. Сравните сложности задач дискретного логарифмирования на эллиптической кривой и в конечном поле.

2. Оцените срок безопасной эксплуатации криптосистемы, основанной на задаче дискретного логарифмирования в группе точек эллиптической кривой, по отношению к методу Полларда, если ха-

рактеристика поля и порядок группы имеют длину 256 бит. Оцените срок действия ключа в начале и в конце срока эксплуатации.

3. Сформулируйте требования к параметрам эллиптической кривой для обеспечения стойкости к методу Полига–Хеллмана.

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, которая, в соответствии с вариантом, реализует один из методов дискретного логарифмирования на эллиптической кривой: метод Полларда или метод Полига–Хеллмана. При реализации алгоритма Полига–Хеллмана допускается использование готовых процедур для вычисления логарифма $log_{P_0} Q_k$. В качестве входных данных программа **П-1** должна принимать параметры эллиптической кривой, образующую точку $P$, порядок $q$ группы $\langle P \rangle$; на выходе – возвращать значение логарифма $d$.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) описание реализованного метода, используемые параметры;

б) время работы программы;

в) число итераций;

г) при результативном завершении работы − вычисленное значение логарифма, в противном случае – промежуточные результаты и расчетное время, оставшееся до завершения работы (через оценку сложности алгоритма).

3. Ответы на контрольные вопросы.
4. Выводы по работе.
5. Листинг программы.

*ЛАБОРАТОРНАЯ РАБОТА № 8*

# РАЗЛОЖЕНИЕ ЧИСЛА НА МНОЖИТЕЛИ НА ЭЛЛИПТИЧЕСКОЙ КРИВОЙ

Цель работы – реализация метода разложения числа на множители с использованием эллиптических кривых.

## Теоретические сведения

Задача разложения на множители числа $n$ над кольцом $\mathbb{Z}$ сводится к нахождению всех простых делителей числа $n$. Методы разложения числа на множители можно разделить на специальные и универсальные.

Специальные методы эффективны для некоторых классов составных чисел и, как правило, используют генератор случайных чисел или псевдослучайное отображение. К таким методам относятся $\rho$-метод Полларда, $(p-1)$-метод Полларда.

Универсальные методы не предполагают никакой информации о виде числа $n$ и его делителях. К таким методам относятся метод пробного деления, метод «giant step – baby step», метод Ферма, метод диофантовой аппроксимации, метод непрерывных дробей, метод квадратичного решета и др.

В 1987 г. Х. Ленстра предложил алгоритм разложения числа на множители с использованием эллиптических кривых.

### 1. Разложение на эллиптической кривой

Метод Ленстры является аналогом $(p-1)$-метода Полларда в группе точек эллиптической кривой, который позволяет найти нетривиальный делитель заданного числа $n$.

Предположим, что число $n$ раскладывается на множители: $n = pq$, где $p \neq q$ – простые числа. Тогда по китайской теореме об остатках имеет место изоморфизм групп:

$$E(\mathbb{Z}/n\mathbb{Z}) \cong E(\mathbb{Z}/p\mathbb{Z}) \oplus E(\mathbb{Z}/q\mathbb{Z}).$$

Умножение любого элемента группы $E(\mathbb{Z}/n\mathbb{Z})$ на $m \in \mathbb{Z}$ сохраняет данный изоморфизм:

$$mE(\mathbb{Z}/n\mathbb{Z}) \cong mE(\mathbb{Z}/p\mathbb{Z}) \oplus mE(\mathbb{Z}/q\mathbb{Z}).$$

В формуле сложения двух точек $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ выражение для углового коэффициента имеет вид: $\lambda = \frac{3x_1^2+A}{2y_1}$, а в формуле удвоения ($P_1 = P_2$) коэффициент соответственно равен $\lambda = \frac{y_2-y_1}{x_2-x_1}$. В некоторых случаях может получиться так, что по модулю $p$ точки различны, а по модулю $q$ одинаковы. Тогда формулы сложения точек дадут неправильный результат, и сумма точек может не лежать на исходной кривой.

Данная ситуация легко обнаруживается вычислением значения $d = \text{НОД}(n, \lambda_1)$, где $\lambda_1$ – знаменатель в формулах углового коэффициента $\lambda$. Если точки заданы в проективных координатах $(X, Y, Z)$, то при вычислении $d$ можно использовать значение координаты $Z$: $d = \text{НОД}(n, Z)$. При $d$, отличном от 1 и $n$, задача нахождения делителя числа $n$ решена.

Указанная ситуация означает, что одним из компонентов разложения точки эллиптической кривой $E(\mathbb{Z}/n\mathbb{Z})$ в прямую сумму является бесконечно удаленная точка, которая при дальнейшем умножении на любое число остается неподвижной.

Пусть $E(\mathbb{Z}/n\mathbb{Z})$ – эллиптическая кривая и $Q$ – точка на ней. Нахождение точки требует вычисления квадратного корня, поэтому можно сначала выбрать точку $Q = (x_0, y_0)$, а потом получить уравнение кривой (сгенерировать коэффициент $A$, тогда значение $B = y_0^2 - x_0^3 - Ax_0 (mod\, n)$). Пусть $D$ – база разложения, содержащая простые числа $p_i$, меньшие заданной границы $m$. Найдем целочисленное произведение $k = \prod_{p_i \in D} p_i^{\alpha_i}$ всех элементов базы $D$, где

$$\alpha_i = \left\lfloor 0{,}5 \frac{log\, n}{log\, p_i} \right\rfloor.$$

Эвристически оптимальный размер базы $D$ связан с вероятностью $u^{-u}$, где $u \approx \frac{ln\, n}{ln\, m}, m$ – порядковый номер наибольшего элемента базы $D$.

**Алгоритм 8.1.** Разложение числа на эллиптической кривой

*Вход*. Число $n$, размер $m$ базы $D$.

*Выход*. Нетривиальный делитель $d$ числа $n$.

1. Выбрать случайную эллиптическую кривую $E(\mathbb{Z}/n\mathbb{Z})$ и точку $Q$ на ней.
2. Положить $i \leftarrow 0, Q_i \leftarrow Q$.
3. При $i > m$ вернуться на шаг 1. В противном случае найти $i$-е простое число $p_i$.
4. Положить $i \leftarrow i + 1, \alpha_i = \left\lfloor 0{,}5 \frac{log\, n}{log\, p_i} \right\rfloor, j \leftarrow 0$.
5. При $j > \alpha_i$ перейти на шаг 3. В противном случае выполнить следующие действия.
   5.1. Положить $Q_i \leftarrow p_i Q_i$. При каждом сложении точек вычислять $d = \text{НОД}(n, \lambda_1)$. При $1 < d < n$ результат: $d$.
   5.2. Положить $j \leftarrow j + 1$ и вернуться на шаг 5.
6. Если нетривиальный делитель числа $n$ не найден, то вернуться на шаг 1. ■

Асимптотическая сложность алгоритма составляет$O(exp(\,(1 + o(1))\sqrt{ln\, n\, lnln\, n}))$, если наименьший из простых делителей числа $n$ имеет порядок $O(\sqrt{n})$.

Этот алгоритм не требует создания базы данных, так как достаточно просто находить очередной элемент базы разложения $D$ и умножать на него текущую точку. Поэтому здесь возможно неограниченное распараллеливание с помощью невзаимодействующих компьютеров, каждый из которых работает со своим семейством эллиптических кривых. Однако разложение при этом ускоряется незначительно.

**Пример 8.1.** Разложение на эллиптической кривой

Найдем делитель числа $n = 661643$. Выбираем эллиптическую кривую $y^2 = x^3 - x + 3231$ и точку $Q = (87, 2)$ на ней. Умножаем точку последовательно на $2^9, 3^6, 5^4, 7^3, 11^2$ и т. д., вычисляя наибольший общий делитель знаменателя углового коэффициента и числа $n$.

Получаем:

$Q_2 = 2^9 Q = (196083,\ 134895)$;

$Q_3 = 3^6 Q_2 = (470021, 282574);$

$Q_5 = 5^4 Q_3 = (546729, 237268);$

$Q_7 = 7^3 Q_5 = (419723, 447921).$

При первом умножении точки $Q_7$ на 11 получаем знаменатель углового коэффициента, равный 90347. Находим $\text{НОД}(n, 90347) = 541$, т.е. $n = 541 \cdot 1223$. ■

Среди составных чисел одинаковой длины быстрее раскладываются те, которые имеют простой делитель меньшего размера.

## Контрольные вопросы

1. Как зависит сложность разложения составного числа заданной длины методом эллиптических кривых от числа различных простых делителей числа $n$?

2. Сравните сложность разложения на эллиптической кривой составного числа вида $n = p^2 q$, где $p, q$ – различные простые числа, и составного числа такой же длины, состоящего из двух различных простых делителей.

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, которая находит разложение составного числа $n$ на эллиптической кривой. В качестве входных данных программа **П-1** должна принимать число $n$, размер $m$ базы $D$; на выходе – возвращать нетривиальный делитель $d$ числа $n$.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) параметры алгоритма (коэффициенты эллиптической кривой, выбранная точка на кривой);

б) время работы программы;

в) число итераций;

г) в случае результативного завершения работы программы – найденное разложение числа $n$ с указанием длины его множителей, в противном случае – промежуточные результаты и расчетное время, оставшееся до завершения работы (через оценку сложности алгоритма).

3. Ответы на контрольные вопросы.
4. Выводы по работе.
5. Листинги программ.

*ЛАБОРАТОРНАЯ РАБОТА № 9*

# АТАКА НА КРИПТОСИСТЕМУ ECIES

Цель работы – изучение атаки на схему гибридного шифрования ECIES, возможной в результате отсутствия проверки принадлежности точки кривой.

## Теоретические сведения

Среди множества атак на криптосистемы на эллиптических кривых существует класс атак, связанных с ошибками при практической реализации. Примером подобной атаки является атака «некорректной кривой» (от англ. Invalid-Curve Attack), возникающая в связи с отсутствием проверки принадлежности точки заданной кривой (Antipa A. et al. Validation of elliptic curve public keys //International workshop on public key cryptography. – Springer, Berlin, Heidelberg, 2003. – С. 211-223.).

### 1. Схема шифрования ECIES

Пусть есть эллиптическая кривая, заданная в краткой форме Вейерштрасса: $E(\mathbb{F}_p): y^2 = x^3 + ax + b, \#E(\mathbb{F}_p) = n \cdot q$. Как правило, в криптографии используются кривые, порядок которых имеет большой простой делитель $q$: для таких кривых задача дискретного логарифмирования является вычислительно трудной.

Гибридный алгоритм шифрования на эллиптических кривых (Elliptic Curve Integrated Encryption Scheme) использует следующие криптографические примитивы:

- функцию формирования ключа KDF;
- алгоритм симметричного шифрования SYM;
- код аутентификации сообщения MAC.

В основе безопасности лежит задача дискретного логарифмирования в группе точек эллиптической кривой. Пусть $P$ – образующая подгруппы группы точек большого простого порядка $q$, $Q = dP$ – открытый ключ, $d -$ закрытый ключ.

**Алгоритм 9.1.** Зашифрование сообщения с помощью ECIES

*Вход*. Сообщение $m$, открытый ключ $Q$.

*Выход*. Шифртекст $(R, c, t)$.

1. Выбрать случайное $r \in \mathbb{Z}: 0 < r < q$.
2. Вычислить $R \leftarrow rP, K \leftarrow rQ$.
3. Сформировать ключи $(k_1, k_2) \leftarrow KDF(K_x)$, где $K_x - x$-координата точки $K$.
4. Вычислить $c \leftarrow SYM_{k_1}(m), t \leftarrow MAC_{k_2}(c)$.
5. Шифртекст $(R, c, t)$. ■

**Алгоритм 9.2.** Расшифрование сообщения с помощью ECIES

*Вход*. Шифртекст $(R, c, t)$, закрытый ключ $d$.

*Выход*. Сообщение $m$.

1. Вычислить $K \leftarrow dR$.
2. Сформировать ключи $(k_1, k_2) \leftarrow KDF(K_x)$.
3. Вычислить $t' \leftarrow MAC_{k_2}(c)$. Если $t' \neq t$, то отбросить шифртекст как некорректный.
4. Расшифровать сообщение $m \leftarrow SYM_{k_1}(c)$.■

Как видно из процедуры зашифрования (алгоритм 9.1), шифртекст содержит точку $R$, которая должна принадлежать исходной кривой $E$. Однако, если на стороне получателя не осуществляется данная проверка, это позволяет выполнить вскрытие закрытого ключа за достаточно небольшой промежуток времени.

## 2. Подготовка данных для атаки

В основе данной атаки «некорректной кривой лежит тот факт, что формулы удвоения и сложения точек на эллиптической кривой не зависят от значения коэффициента $b$.

Если при вычислениях не осуществляется проверка принадлежности точки эллиптической кривой, то в таком случае атакующий может посылать точки, принадлежащие другим кривым, с тем же значением коэффициента $a$, но другим коэффициентом $b$. Для проведения атаки необходимо, чтобы порядок таких кривых был сильно составным числом (что подразумевает достаточно быстрое нахождение

дискретного логарифма). Например, атакующий послал точку малого порядка $P_{inv} \in E_{inv}$, где кривая $E_{inv}: y^2 = x^3 + ax + b'$ отличается от исходной значением свободного члена $b'$, и получил в качестве результата точку $Q_{inv} = dP_{inv}$, где $d$ – закрытый ключ. Так как порядок группы точек кривой $E_{inv}$ является сильно составным, а точка $P_{inv}$ имеет малый порядок, то по полученному значению $Q_{inv}$ можно вычислить значение $d$ по модулю порядка точки $P_{inv}$.

**Пример 9.1.** Подготовка данных для атаки

Пусть для зашифрования используется стандартная кривая secp256r1 (prime256v1), имеющая параметры:

```
a= 0xffffffff00000001000000000000000000000000fffffffffffffffffffffffc
b= 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
p=0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff
```

Данная кривая имеет простое число точек, т.е. ее порядок – большое простое число, таким образом, задача дискретного логарифмирования в группе ее точек является «сложной».

Для проведения атаки сначала необходимо сгенерировать кривые, отличающиеся от исходной коэффициентом $b$ и имеющие сильно составной порядок группы точек. Примеры кривых:

```
b = 3
Число точек: 2^4 * 3 * 7 * 13 * 37 * 97 * 113 *
6536486346623006500929960951851831892127909863212781930170424846583 3
b = 6
Число точек: 2^2 * 5 * 4003 * 16033 * 102001 *
884390526899024919222120216398419141830968591728388637593606029
b = 18
Число точек: 2^3 * 7 * 29 * 349 * 13998374731216357 *
145945202271086275094607874968851530052467149806442207 73
```

Кривая с коэффициентом $b = 3$ имеет сильно составной порядок, поэтому она может быть использована для реализации атаки и нахождения значения закрытого ключа по модулю 3, 7, 13, 37, 97, 113. Кривая с коэффициентом $b = 6$ может быть использована для нахождения значения по модулю 5, кривая с $b = 18$ – для нахождения значения по модулю 29, 349, и т.п.

После отбора кривых необходимо сгенерировать точки соответствующего порядка. Пример отобранных для проведения атаки значений (в виде $(b, l, R_{inv})$):

```
[(1, 71, (367361742554304813224642348745565282452036758593994133061088231320706842137 26 :
712166368413282124386734971394782872431033932961606197450257735625186139538 88)),
(3, 7, (868619487793207237358094091663386610139248700735331534014053803635912856094 45 :
929590933807737392392568436727964912958272795258977388760182166040590034339 08)),
```

(3, 13, (10303641122814403202831797928901783793501173061388314454812693874640936831193 5 :
484893757433981059005306886495648836923876042606444428693285333480081139532 60)),
(3, 37, (68293383902662374822833102466651033088912283363194773184796445222605560563 09 :
41090420942805220562393108111996992853311669414049431247759332665206023742070)),
(4, 179, (83229040458127123746763986781924855834855773376892353224964588596517159218446 :
89598705492901877225454412430249165727296653092812945717144067107278708102557)),
(6, 5, (39750170337003670022494616352462194843889127399495951655079750874966815322615 :
65141808569782695839169856395194564992307712545138211871768272355352435143036)),
(10, 17, (28441888532776736443884565435952912890996156064275160816382945068056727079366 :
62774884647663948051590459932614506163828928024892425144614982360172294976753)).

Генерируя такие точки с малым порядком, можно найти значение закрытого ключа $d$ по различным модулям, а затем восстановить значение ключа с помощью Китайской теоремы об остатках. ■

## 3. Атака на ECIES, связанная с отсутствием проверки принадлежности точки кривой

В схеме ECIES атака «некорректной кривой» возможна в случае, если получатель шифртекста не осуществляет проверку принадлежности точки $R$ кривой, указанной в параметрах. Основная сложность в проведении атаки заключается в прохождении проверки значений MAC-тега $t$ на шаге 3 алгоритма 9.2. Для этого необходимо «подтвердить» знание соответствующей для $R$ точки $K$, создав корректный MAC-тег $t$.

Для атаки необходимо получить шифртекст произвольного сообщения и выполнить следующие действия:

– точку $R$ заменить на точку $R_{inv}$ малого простого порядка $l$ (см. Пример 9.1);

– сгенерировать ключ для тега, вычислив KDF от $x$-координаты точки $P_{inv}$ и взяв в качестве ключа вторые 32 байта (первые 32 байта соответствуют ключу используемого симметричного алгоритма SYM):

– вычислить новый тег $t_{inv}$ на основе нового ключа и заменить им имеющийся в шифртексте тег.

Измененный шифртекст $(R_{inv}, c, t_{inv})$, пройдет проверку на шаге 3 алгоритма 9.2 и будет принят получателем, если тег, вычисленный с помощью ключа от $x$-координаты точки $K' = dR_{inv}$, где $R_{inv} -$ точка из измененного шифртекста, $d -$ закрытый ключ, будет соответствовать тегу из присланного атакующим шифртекста. Это возможно только в том случае, если точки $K = d_l R_{inv}$ и $K' = dR_{inv}$ совпадут, т.е. $d_l \equiv \pm d (mod\ l)$.

**Пример 9.2.** Нахождение закрытого ключа по произвольному модулю $d_l \equiv d(mod\ l)$

Пусть есть точка

```
R_inv=(397501703370036700224946163524621948438891273994959516550797508749668
815322615,651418085697826958391698563951945649923077125451382118717682723553
52435143036),
```

которая имеет порядок 5 на кривой с коэффициентом $b = 6$.

Вычислим различные $d_l \in [0, l-1]$ и соответствующие точки $K = d_l R_{inv}$:

```
l =  0
K: (0 : 1 : 0)
l =  1
K: (397501703370036700224946163524621948438891273994959516550797508749668153226
15 : 651418085697826958391698563951945649923077125451382118717682723553524351430
36)
l =  2
K: (675857544816761447415602744025507216949703490043640780965211656403653257965
02 : 492329777383347454259863371257048797229448752343914876166380258810455093989
45)
l =  3
K: (675857544816761447415602744025507216949703490043640780965211656403653257965
02 : 665591114720215033367111098237026938071412681808988265788956054278215884550
06)
l =  4
K: (397501703370036700224946163524621948438891273994959516550797508749668153226
15 : 506502806405735529235275905542130085377784308701521023237653589535146627109
15)
```

Пусть закрытый ключ получателя равен $d = 1234568$. При расшифровании получатель вычисляет точку

$K' = dR_{inv}$ = (675857544816761447415602744025507216949703490043640780965211656403653257965 02 : 665591114720215033367111098237026938071412681808988265788956054278215884550 06 ),

которая совпадет с $K = d_l R_{inv}$, когда $d_l = 2$ и $d_l = 3$, т.е. $d_l \equiv \pm 3\ (mod\ 5)$. В этом случае тег из присланного шифртекста совпадет с тегом, вычисленным получателем на основе ключа из точки $K'$. В результате, алгоритм вернет расшифрованное сообщение (расшифрование произойдет некорректно, но для реализации атаки важен сам факт того, что прошла проверка тега). Из этого можно сделать вывод, что значение закрытого ключа $d = d_l \equiv \pm 3\ (mod\ 5)$. Действительно, $1234568 \equiv 3\ mod\ 5$, и значение ключа по модулю 5 было найдено. ■

Таким образом, вычисляя значения $d_{l_i} \equiv d(mod\ l_i)$, можно восстановить полное значение закрытого ключа $d$. Минимальное количество $l_i$ определяется из соотношения: $\prod l_i \geq$ threshold, где порог threshold определяет значение для однозначного восстановления ключа по КТО.

**Алгоритм 9.3.** Атака «некорректной кривой» на схему ECIES

*Вход*. Параметры схемы ECIES.

*Выход*. Значение закрытого ключа $d$.

1. Подготовить данные для атаки (см. п. 2). Сгенерировать список из элементов $\left(b_i, l_i, R_{inv_i}\right), \prod l_i >$ threshold. Получить шифртекст $(R, c, t)$ произвольного файла.

2. Для каждого элемента $(b_i, l_i, R_{inv_i})$, где $b_i$- коэффициент некорректной кривой $E_{inv_i}$, $l_i$, – малый простой порядок точки $R_{inv_i} \in E_{inv_i}$, сама точка $R_{inv_i}$, выполнить:

2.1. Заменить в шифртексте точку $R$ на точку $R_{inv_i}$.

2.2. Для $j$ от 1 до $l_i - 1$:

2.2.1. Вычислить точку $K = j * R_{inv_i}$.

2.2.2. На основе $x$-координаты $K$ сгенерировать ключи:

$$(key_{SYM}, key_{MAC}) \leftarrow KDF(K_x).$$

2.2.3. Вычислить тег $\hat{t}$ на основе ключа $key_{MAC}$ с помощью MAC.

2.2.4. Заменить в шифртексте тег $t$ тег $\widehat{t_j}$.

2.2.5. Отправить сформированный шифртекст $\left(R_{inv_i}, c, \widehat{t_j}\right)$ на расшифрование. Если отправитель вернул расшифрованный файл (т.е. проверка тега MAC прошла успешно), то положить $d_{l_i} \equiv \pm j \ (mod\ l_i)$ и перейти на шаг 2.

2.2.6. Если для всех $j$ от 1 до $l_i - 1$ отправитель отбросил шифртекст как некорректный, то положить $d_{l_i} \equiv 0 \ (mod\ l_i)$ и перейти на шаг 2.

3. С помощью Китайской теоремы об остатках выполнить восстановление значение закрытого ключа по полученным значениям $d_{l_i}$. ■

## Контрольные вопросы

1. Сформулируйте методы противодействия атакам, связанным с некорректной реализацией криптографии на эллиптических кривых.

2. Приведите способ реализации подобной атаки в случае схемы установления ключа Диффи-Хеллмана на эллиптических кривых.

3. Привести пример программных продуктов/библиотек, в которых была обнаружена уязвимость, позволяющая осуществить атаку «некорректной кривой».

## Порядок выполнения работы

Получить у преподавателя вариант задания и разработать программу **П-1**, которая выполняет атаку на некорректную реализацию ECIES.

Предоставленный вариант представляет собой адрес сервера, на котором запущен алгоритм шифрования ECIES на основе эллиптических кривых. В качестве входных данных программа **П-1** должна принимать адрес сервера; на выходе – возвращать закрытый ключ сервера.

## Содержание отчета

1. Формулировка задания.
2. Выполненная работа:

а) параметры схемы ECIES, используемые сервером;

б) сгенерированные для атаки данные;

в) найденное в результате атаки значение закрытого ключа сервера.

3. Ответы на контрольные вопросы.
4. Выводы по работе.
5. Листинги программ.

# ОПИСАНИЕ ФОРМАТА ASN.1

Любые данные в ASN.1 представляются в виде последовательности «Тэг – Длина – Значение». При этом «Тэг» определяет тип данных и имеет длину 1 байт, а «Длина» задает размер следующего поля «Значение». В ASN.1 предусмотрены как базовые типы, приведенные в таблице А.1, так и сложные (составные) типы данных, приведенные в таблице А.2.

Таблица А.1 – Базовые типы данных в ASN.1

| Код | Тип | Комментарий |
|---|---|---|
| 0x02 | INTEGER | Целое число |
| 0x0C | UTF_STRING | Строка в кодировке UTF-8 |
| 0x04 | BYTE_STRING | Последовательность байтов |
| … | … | … |

Таблица А.2 – Сложные (составные) типы данных в ASN.1

| Код | Тип | Комментарий |
|---|---|---|
| 0x30 | SEQUENCE | Последовательность |
| 0x31 | SET | Строка в кодировке UTF-8 |
| … | … | … |

В значениях базовых типов записываются элементарные данные. В значениях составных типов могут присутствовать только другие поля. Причем, в SEQUENCE – поля любых типов, однако обычно в известном порядке, а в SET – только одного и того же типа, а порядок следования элементов не имеет значения.

## 1. Сохранение длины полей

Сохраняемые значения не имеют ограничений по длине (теоретическое ограничение – $256^{127}$ байт). Поле «Длина» имеет определенный формат:

1) если длина меньше 0x80 (128) байтов, тогда в поле «Длина» записывается число в бинарном виде без изменений;

2) если длина больше или равна 128 байтам, тогда в первом байте в младших семи битах сохраняется длина следующих байтов поля «Длина», а старший бит выставляется в единицу. Следующие байты, количество которых определено первым, задают непосредственно длину поля данных. Порядок байтов – MSB.

Разбор поля «Длина» производится следующим образом. Сначала проверяется старший бит старшего байта $((b\&0x80) == 0)$, & – побитовое И). Если он нулевой, длина менее 128 байт и сохранена в первом байте, то общая длина поля «Длина» – 1 байт.

Если старший байт единичный $((b\&0x80)! = 0)$, тогда применен префикс поля «Длина». Информация сохранена в последующих $k$ байтах, причем $k = b\&0x7F$ (младшие 7 битов). Считываются следующие $k$ байтов и преобразуются в число. Обычно $k$ не более 2–3.

## 2. Примеры представления ключей RSA

Для сохранения структуры предназначен тип SEQUENCE. Он сохраняет последовательность любых базовых и составных типов, в частности может включать в себя другие последовательности.

Обычно ключ RSA сохраняют как последовательность модуля и экспоненты, каждый из которых является целым числом.

Рассмотрим пример. Пусть сохраняется ключ RSA

$e = 3$,

$n = (11317E789C45CCCC7436C384D4354945)_{16}$.

Первый элемент последовательности – модуль $n$. В самом числе $n$ старший бит старшего байта равен 0, поэтому дописывание нулем не требуется. Представление числа $n$ имеет длину 16 байтов, которая может сохраняться без префикса. Поэтому ASN.1-представление числа $n$ будет иметь вид

0x02 10 11317E789C45CCCC7436C384D4354945,

а именно:

| | TAG | LEN | Value | |
|---|---|---|---|---|
| | 0x02 | 0x10 | 11317E ….4945 | |

Представление числа 3 также без дописывания нулями и без префикса длины: 02013

| | TAG | LEN | Value |
|---|---|---|---|
| | 0x02 | 0x01 | 0x03 |

Последовательность имеет тэг 0x30. Длина первого представления составляет 18 байт, второго – 3 байта. Общая длина данных последовательности – 21 байт = 0x15 – также может быть записана без префикса длины.

| TAG | LEN | TAG | LEN | Value | TAG | LEN | Value |
|---|---|---|---|---|---|---|---|
| 0x30 | 0x15 | Элемент 1 | | | Элемент 2 | | |
| | | 0x02 | 0x10 | 11317E….4945 | 0x02 | 0x01 | 0x03 |

Полученный общий результат будет равен

3015 0210 11317E789C45CCCC7436C384D4354945 020103.

К достоинствам описанного представления стоит отнести универсальность, стандартизованность и заложенные возможности по определению ошибок. К недостаткам – необходимость привлечения дополнительных вычислительных мощностей для обработки данных перед отправкой или после получения, избыточность данных.

Описанный формат хранения ключей RSA является достаточно распространенным при сохранении данных ключа в сертификатах, а также в запросах и ответах аппаратных модулей безопасности, работающих с RSA.

## 3. Формат зашифрованного файла

Файл состоит из заголовка и зашифрованной при помощи симметричного алгоритма шифрования последовательности данных.

Заголовок представляет собой ASN.1-последовательность ключей и данных о файле. Первый символ файла – 0x30, далее длина всего заголовка и данные. Содержимое заголовка приведено в таблице А.3.

Множество ключей представляет собой ASN.1-множество, элементами которого являются последовательности с данными об использованных ключах. Каждая последовательность представляет собой открытый ключ и соответствующий ему шифртекст. Описание последовательностей приведено в таблице А.4. Дополнительные данные приведены в таблице А.5.

Таблица А.3 – Содержимое заголовка

| Поле | Тэг | Комментарии | Значение элемента |
|---|---|---|---|
| Множество ключей | 31 | Данные заголовка об использованных ключах | 0x31 $len\_keys$ $key1\_seq$, … $keyN\_seq$ |
| Открытые данные о файле (сообщении) | 30 | Дополнительные данные о файле (сообщении). Опционально | 0x30 0x00 |

Таблица А.4 – Описание последовательностей

| Поле | Тэг | Комментарии | Значение элемента |
|---|---|---|---|
| Идентификатор алгоритма шифрования | 04 | 0x0001 – RSA | 0x04 0x02 0x00 0x00 |
| Строковый идентификатор ключа, псевдоним | 0C | Опционально – строковый идентификатор ключа. Если не указан, следует использовать пустую строку 0x0C, 0x00 | 0x0C $len\_i$ UTF8(id) |
| Открытый ключ | 30 | Для RSA – последовательность модуля и экспоненты | 0x30,$len\_s$, 0x02, $len\_n$ , $n$, 0x02, $len\_e$, $e$ |
| Параметры алгоритма | 30 | Для RSA не используется, общий параметр $n$ передается внутри открытого ключа | 0x30 0x00 |
| Шифртекст алгоритма с открытым ключом (зашифрованный проме- | 30 | Для RSA – последовательность, состоящая из одного числа $c$. | 0x30,$len\_s$, 0x02, $len\_c$, $c$ |

| Поле | Тэг | Комментарии | Значение элемента |
|---|---|---|---|
| жуточный ключ симметричного алгоритма) | | Если ключ не указан, можно использовать последовательность нулевой длины: 0x30, 0x00 | |

где,

$n$ – модуль RSA длины $len\_n$;

$e$ – открытая экспонента RSA длины $len\_e$;

$c$ – шифртекст алгоритма RSA длины $len\_c$;

$len\_s$ – длины последовательностей.

Таблица А.5 – Дополнительные данные

| Поле | Тэг | Комментарии | Значение элемента |
|---|---|---|---|
| Идентификатор алгоритма шифрования | 04 | Идентификатор симметричного алгоритма<br>0x0121 – DES в режиме простой замены<br>0x0132 – Triple-DES в режиме простой замены<br>0x0133 – Triple-DES в режиме CBC<br>0x1081 – AES в режиме простой замены<br>0x1082 – AES в режиме CBC | 0x04<br>$len\_fclfile\_content\_len$ |

| Поле | Тэг | Комментарии | Значение элемента |
|---|---|---|---|
| Длина сообщения | 02 | Опционально – длина передаваемого сообщения (файла). Если длина не указана, следует использовать пустую строку 0x02, 0x00. Используется для выравнивания | 0x0C $len\_fclfile\_content\_len$ |

Заголовок с данными:

303F31323030040200010C0474657374301502101131 7E789C45CCCC7436C384D435494502010330003 00B02090A0B0C0A0B0C0A0B0C3009040201210203133870

разбирается как ASN.1-структура следующего вида:

Start Tag Len

0 30 63 : SEQUENCE { − заголовок
2 31 50 : SET { − множество ключей
4 30 48 : SEQUENCE { − первый ключ
6 04 2 : OCTET STRING – идентификатор алгоритма (RSA)
: 00 01
10 0C 4 : UTF8String 'test'
16 30 21 : SEQUENCE { − значение открытого ключа
18 02 16 : INTEGER − модуль, число $n$
: 11 31 7E 78 9C 45 CC CC 74 36 C3 84 D4 35 49 45
36 02 1 : INTEGER 3 – открытая экспонента, число $e$
: }
39 30 0 : SEQUENCE {} – параметры криптосистемы
41 30 11 : SEQUENCE { − шифртекст
43 02 9 : INTEGER – число $c$

```
                :          0A 0B 0C 0A 0B 0C 0A 0B 0C
                :        }
                :      }
                :    }
        54 30   9 :  SEQUENCE {
        56 04   2 :    OCTET STRING
                :      01 32 − идентификатор симметричного алгорит-
ма, Triple-DES
        60 02   3 :    INTEGER 1259632 – длина файла
                :    }
                :  }
```

## 4. Формат файла подписи

Формат файла подписи совпадает с форматом файла сообщения, однако файл подписи не содержит в себе тела сообщения. Дополнительные данные о файле также могут быть пропущены, поскольку для подписи не несут никакой важной информации.

Пример файла подписи:

3041313D303B040200060C08746573745369676E30150210113 17E789C45CCCC7436C384D435494502010330003012021009F505B178C4316B0BDAE8C0162254F53000

```
 0 30  65 : SEQUENCE {– заголовок
 2 31  61 :  SET { – множество ключей (обычно только 1)
 4 30  59 :   SEQUENCE { – ключ и подпись
 6 04   2 :    OCTET STRING
               00 06 – идентификатор алгоритма (RSA-SHA1)
10 0C  8 :    UTF8String 'testSign' – строковый идентификатор
ключа
20 30  21 :    SEQUENCE {
22 02  16 :     INTEGER – модуль, число n
                11 31 7E 78 9C 45 CC CC 74 36 C3 84 D4 35 49
45
40 02   1 :     INTEGER 3 – открытая экспонента, число e
                }
43 30   0 :    SEQUENCE {}
```

```
   45 30  18 :       SEQUENCE { – подпись сообщения
   47 02  16 :         INTEGER – число s
             :          09 F5 05 B1 78 C4 31 6B 0B DA E8 C0 16 22 54
F5
             :        }
             :       }
             :     }
   65 30   0 :   SEQUENCE {} – дополнительных данных нет
             :   }
```

Наряду с алгоритмом подписи RSA-SHA1 допустимо использовать алгоритм RSA-SHA256, который использует в качестве хэш-функции алгоритм SHA-256. Идентификатор алгоритма – 0x0040.

*Приложение Б*

# РАБОТА С УТИЛИТОЙ CRYPTOOL

CrypTool (www.cryptool.org) – открытый образовательный проект, направленный на изучение и анализ различных криптографических алгоритмов. CrypTool поддерживает визуализацию современных криптографических примитивов, генерацию параметров, методы криптоанализа. В рамках проекта существует несколько утилит, которые отличаются функционалом:

- CrypTool 1 – версия, предназначенная для операционных систем семейства Windows и включающая в себя большинство современных криптографических примитивов, а также визуализацию теоретико-числовых алгоритмов;
- версия CrypTool 2 основана на концепции визуального программирования и позволяет пользователю комбинировать различные алгоритмы и функции;
- JCrypTool доступен для платформ на базе Windows, Linux, MacOS. Помимо классических шифров включает в себя некоторые алгоритмы постквантовой криптографии.

Версия CrypTool 1 поддерживает:

- классические шифры: шифр Цезаря, Виженера, Атбаш, Play-Fair и др.;
- современные симметричные шифры: IDEA, RC2, RC4, RC6, DES, 3DES, AES и др.;
- шифрование и подпись RSA, криптографию на эллиптических кривых (в том числе комбинированные схемы RSA-AES, ECC-AES);
- хэш-функции: MD2, MD5, SHA-1, SHA-256, SHA-512, RIPEMD-160 и др.
- генерацию псевдослучайных чисел и их тестирование (тест серий, покер-тест и др.), генерацию ключей;
- base64- и UU-кодирование и др.;
- методы анализа файлов (энтропийный, автокорреляционный анализ, построение гистограмм и др.);

– методы криптоанализа: разложение на множители модуля RSA, атака на RSA с использованием решеток, генерация коллизий хэш-функции и др.;

– образовательная игра «Number Shark»;

– визуализация операции сложения точек на эллиптической кривой, принципов работы алгоритмов AES, DES, шифровальной машины Энигма, учебные материалы по теории чисел и др.

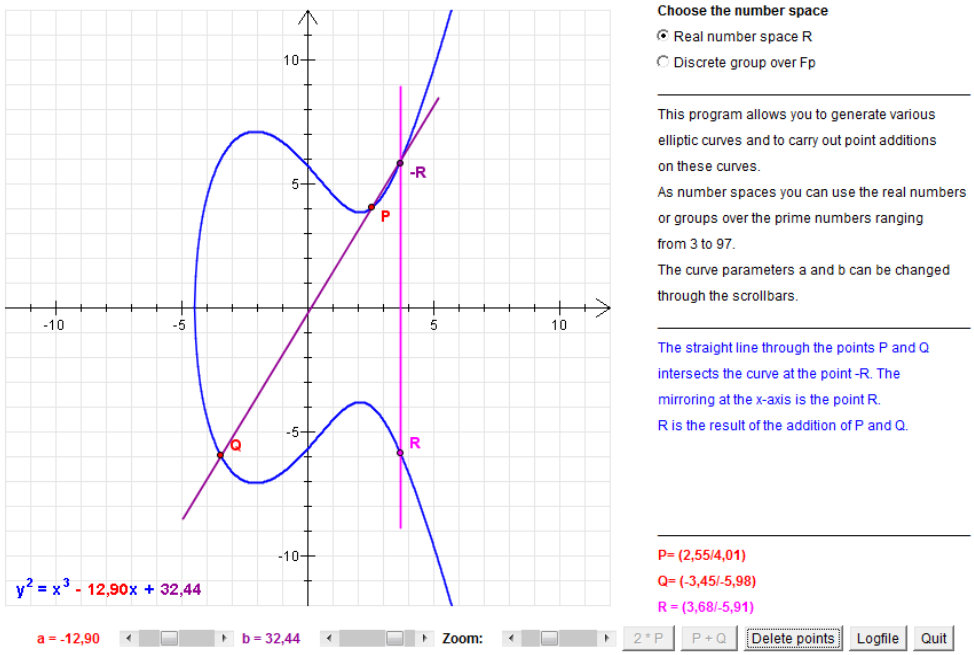


Рисунок Б.1 – Визуализация закона сложения точек на эллиптической кривой в CrypTool 1

Утилита CrypTool 1 предоставляет возможность анализа параметров криптосистемы RSA. Для этого в пункте «Asymmetric Encryption» раздела «Analysis» выбрать «Factorization of a Number», ввести значение модуля $n$ и запустить процедуру тестирования, нажав «Complete factorization into primes».

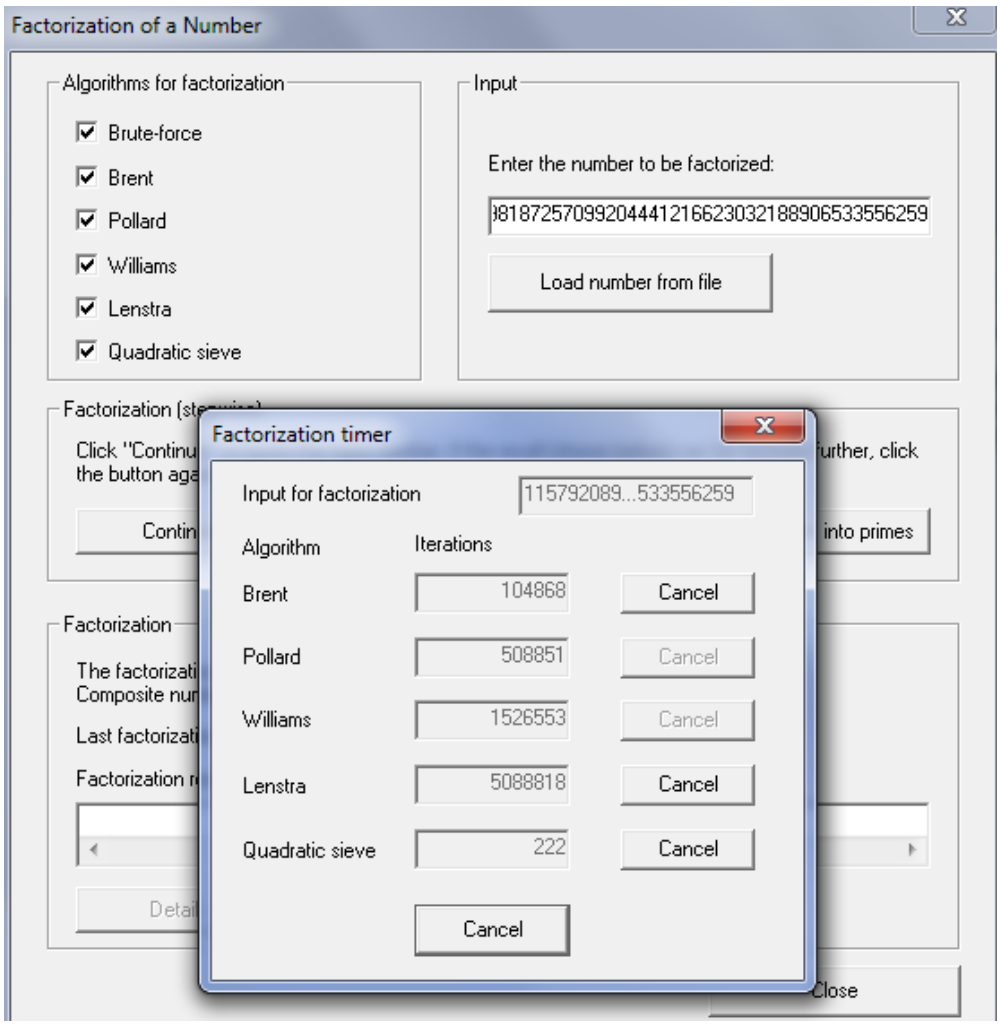


Рисунок Б.2 – Разложение на множители в CrypTool 1

Помимо алгоритмов разложения на множители CrypTool 1 реализует атаку на криптосистему RSA с использованием решеток (раздел «Analysis/Asymmetric Encryption/Lattice-Based Attack on RSA»), которая эффективна при следующих предположениях:

- в случае частично известного открытого текста;
- закрытый показатель $d$ слишком мал по сравнению с $n$;
- частично известен один из множителей числа $n$.

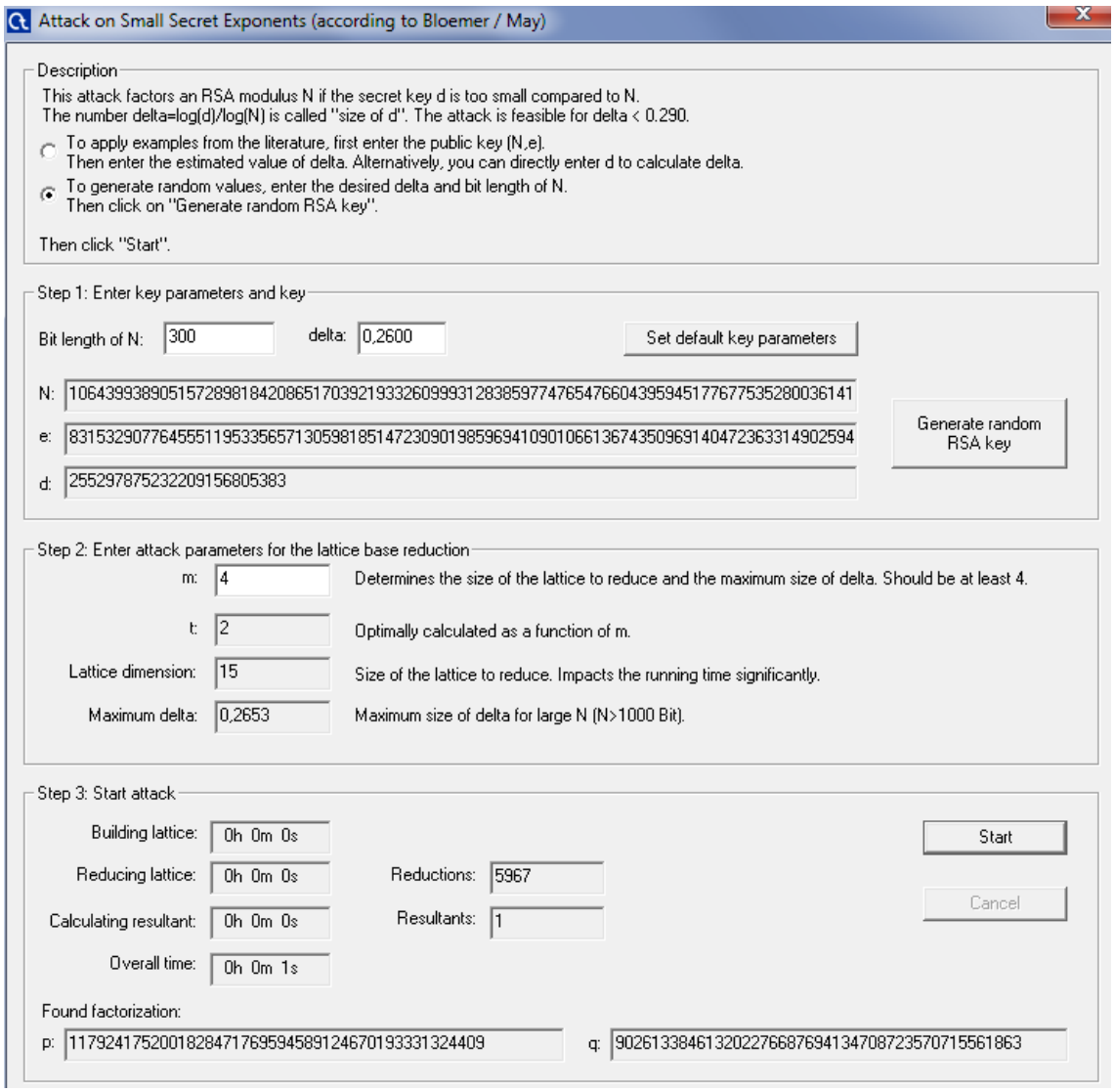


Рисунок Б.3 – Атака на RSA в случае небольшой длины закрытого показателя $d$ в CrypTool 1

Таким образом, утилита CrypTool предоставляет широкий набор инструментов для изучения принципа действия различных криптосистем, визуализации атак, генерации и тестирования параметров.

# ASN.1- ФОРМАТ СООБЩЕНИЙ ДЛЯ КРИПТОСИСТЕМ НА ОСНОВЕ ЗАДАЧИ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ

## 1. Варианты заданий

| Вариант | Протокол/ схема | Идентификатор |
|---|---|---|
| 1 | Протокол шифрования Эль-Гамаля | 0x80010201, $f(m, b^y) = mb^y (\mathrm{mod} p)$ |
| 2 | | 0x80010202, $f(m, b^y) = m + b^y (\mathrm{mod} p)$ |
| 3 | | 0x80010203, $f(m, b^y) = m - b^y (\mathrm{mod} p)$ |
| 4 | Протокол Диффи–Хеллмана | 0x0021 |
| 5 | | |
| 6 | Протокол Месси–Омуры | 0x80070200 |
| 7 | | |
| 8 | Схема подписи Эль-Гамаля | 0x80060200 |
| 9 | | |
| 10 | | |

## 2. Реализация протокола Эль-Гамаля шифрования с открытым ключом

Пусть $G = \mathbb{F}_p^*$, $M = G$. Формат файла для передачи является обобщением формата, используемого для лабораторной работы № 1 и описанного в *Приложении А*. Однако в файле могут быть представлены различные типы ключей.

Шифртекстом является последовательность чисел

ElGamalEncryptedMessage ::= SEQUENCE {

$a^y$– INTEGER,

$c$ – INTEGER

}

Открытым ключом является последовательность

ElGamalPublicKey ::= SEQUENCE {

$b$ – INTEGER – открытый ключ

}

Параметры протокола содержат параметры группы $G = \mathbb{F}_p^*$: характеристику поля и образующую:

ElGamalParameters ::= SEQUENCE {

prime INTEGER, – простое число $p$

generator INTEGER – образующая $a$

}

### 3. Реализация протокола Диффи–Хеллмана

Параметры протокола содержат параметры группы $G = \mathbb{F}_p^*$: характеристику поля и образующую:

DHParameters ::= SEQUENCE {

prime INTEGER, – простое число $p$

generator INTEGER – образующая $a$

}

Сообщения протокола представляются в виде последовательности из одного числа

DHMessage ::= SEQUENCE {

$a^x$ или $a^{xy}$ – INTEGER,

}

Открытый ключ не задействуется и представляет собой пустую последовательность.

Сторона 1. Клиент пытается установить TCP-соединение с сервером.

Клиент генерирует случайный закрытый показатель $x$ и передает на сервер значение $a^x$. Также передаются параметры системы: образующая $a$, число $p$. Таким образом, на сервер передается три параметра ($a^x$,$a$, $p$). Формат сообщения:

SEQUENCE {– заголовок

SET {– множество ключей, 1 задействован

SEQUENCE {– первый ключ

OCTET STRING – идентификатор алгоритма (протокол Диффи–Хеллмана)

00 21

UTF8String 'dh' – может не задействоваться или использоваться для идентификации будущего установленного ключа

SEQUENCE { } – значение открытого ключа, не используется

SEQUENCE {– параметры криптосистемы

INTEGER – простое число $p$

PP PP … PP

INTEGER – образующая $a$

AA AA … AA

}

SEQUENCE {– шифртекст, показатель $a^x$

INTEGER – число $c$

CC CC … CC

}

}

}

SEQUENCE {} – параметры файла, не используются

}

Сторона 2. Сервер ведет ожидание входящих подключений.

При входящем подключении сервер принимает тройку $(a^x, a, p)$, генерирует случайный показатель $y$ и передает клиенту значение $a^y$. Также вычисляется значение $a^{xy}$. Клиенту передается сообщение следующего формата:

SEQUENCE {– заголовок

SET {– множество ключей, 1 задействован

SEQUENCE {– первый ключ

OCTET STRING – идентификатор алгоритма (протокол Диффи–Хеллмана)

00 21

UTF8String 'dh' – может не задействоваться или использоваться для идентификации будущего установленного ключа

SEQUENCE { } – значение открытого ключа, не используются

SEQUENCE { } – параметры криптосистемы

SEQUENCE {– шифртекст

INTEGER – значение $a^{xy}$

CC CC … CC

}

}

}

SEQUENCE {} – параметры файла, не используются

}

Параметры при передаче ответа сервером могут быть пустыми или скопированы из запроса.

Сторона 1. Клиент принимает $a^y$, вычисляет $a^{xy}$.

Из установленного между клиентом и сервером закрытого ключа $a^{xy}$ получается ключ AES: $K = a^{xy} \mathrm{mod} 2^{256}$.

Далее клиент и сервер могут обмениваться сообщениями зашифрованными алгоритмом AES.

Клиент должен быть способен передать короткое текстовое сообщение на сервер. Сообщение имеет следующий формат.

SEQUENCE {– заголовок

SET { } – множество ключей, не задействовано

SEQUENCE {– параметры файла (сообщения)

OCTET STRING

10 82 – идентификатор симметричного алгоритма, AES

INTEGER XXXX – длина файла до выравнивания

}

}

Формат заголовка совпадает с форматом в лабораторной работе № 1, однако секция с открытыми ключами не задействуется.

Тело сообщения дописывается после заголовка. Если длина сообщения не кратна 16 байтам, то открытый текст дополняется символами с кодом 0x03 так, чтобы длина сообщения (файла) перед зашифрованием была кратна 16 байтам.

При передаче необходимо указывать длину открытого текста до зашифрования. Длина содержимого, которое необходимо считать из входного потока TCP-соединения, кратно длине блока симметричного шифра и определяется как наименьшее число, которое превосходит длину сообщения до выравнивания.

Сторона 2. Сервер принимает, расшифровывает и выводит на экран принятое сообщение.

## 4. Реализация протокола бесключевого шифрования Месси–Омуры

Сторона 1. Клиент (отправитель) пытается установить TCP-соединение с сервером.

После успешной установки соединения клиент производит передачу сообщения, вычисляет $t \leftarrow f(m)$ для сообщения $m$. Если $t$ является единичным элементом $e$ группы $G$, то результат: сообщение зашифрованию не подлежит.

Отправитель также генерирует случайный показатель $a$, обратимый по модулю $r$, вычисляет элемент $t^a$ группы $G$ и передает на сервер $t^a$ получателю.

На сервер передаются параметры (однократно, в первом сообщении диалога)

MasseyOmuraParameters ::= SEQUENCE {
prime INTEGER, – простое число $p$
$r$ INTEGER, – порядок $r$ группы
}

и шифртекст

MasseyOmuraMessage1 ::= SEQUENCE {
$t^a$– INTEGER – первый шифртекст, $t^a$
}

Формат сообщения:

SEQUENCE {– заголовок
SET {– множество ключей, один задействован
SEQUENCE {– первый «ключ»
OCTET STRING – идентификатор алгоритма (протокол Месси–Омуры)

80 07 02 00

UTF8String 'mo' – может не задействоваться или использоваться для идентификации будущего переданного сообщения

SEQUENCE { } – значение открытого ключа, не используется

SEQUENCE {– параметры криптосистемы

INTEGER – простое число $p$

PP PP … PP

INTEGER – порядок $r$ группы

RR RR … RR

}

} – параметры криптосистемы

SEQUENCE {– шифртекст

INTEGER – показатель $t^a$

XX XX … XX

}

}

}

SEQUENCE {} – параметры файла, не используются

}

Сторона 2. Сервер ведет ожидание входящих подключений.

При соединении клиента принимает информацию, вырабатывает случайный показатель $b$, обратимый по модулю $r$, вычисляет значение $(t^a)^b = t^{ab}$ и посылает его отправителю.

Клиенту передается шифртекст (параметры в ответ не включаются). Формат сообщения:

SEQUENCE {– заголовок

SET {– множество ключей, один задействован

SEQUENCE {– первый «ключ»

OCTET STRING – идентификатор алгоритма (протокол Месси–Омуры)

80 07 02 00

UTF8String 'mo' – может не задействоваться или использоваться для идентификации будущего переданного сообщения

SEQUENCE { } – значение открытого ключа, не используется

SEQUENCE { } – параметры криптосистемы, не используются

SEQUENCE {– шифртекст

INTEGER – значение $t^{ab}$

XX XX … XX

}

}

}

SEQUENCE {} – параметры файла, не используются

}

Сторона 1. Клиент (отправитель) возводит полученный шифртекст в степень $a^{-1}$: $(t^{ab})^{a^{-1}} = t^b$ и посылает значение $t^b$получателю.

Содержимое сообщения аналогично предыдущему, однако сообщение дополняется результатом шифрования файла при помощи алгоритма AES:

SEQUENCE {– заголовок

SET {– множество ключей, один задействован

SEQUENCE {– первый «ключ»

OCTET STRING – идентификатор алгоритма (протокол Месси–Омуры)

80 07 02 00

UTF8String 'mo' – может не задействоваться или использоваться для идентификации будущего переданного сообщения

SEQUENCE { } – значение открытого ключа, не используется

SEQUENCE { } – параметры криптосистемы, не используются

SEQUENCE {– шифртекст

INTEGER – показатель $t^b$

XX XX … XX

}

}

}

SEQUENCE {

OCTET STRING

10 82– идентификатор симметричного алгоритма, AES

INTEGER XXXX – длина файла до выравнивания

}– параметры файла, не используются

}

Ключ $k = m \mathrm{mod} 2^{256}$ алгоритма AES. Выравнивание сообщения и определение длины исходного сообщения выполняются аналогично вариантам 4, 5.

Сторона 2. Сервер возводит полученный шифртекст в степень $b^{-1}$: $(t^b)^{b^{-1}} = t$ и вычисляет $m \leftarrow f^{-1}(t)$. По известному $m$ вычисляется ключ AES и переданное сообщение.

## 5. Реализация схемы подписи Эль-Гамаля

Формат файла для передачи является обобщением формата, используемого для лабораторной работы № 1 и описанного в *Приложении А*. Однако в файле могут быть представлены различные типы ключей.

Подписью сообщения является последовательность чисел:

ElGamalSign ::= SEQUENCE {

$w$ – INTEGER,

$s$ – INTEGER

}

Ключом является последовательность:

ElGamalSignPublicKey ::= SEQUENCE {

$b$ – INTEGER – открытый ключ

}

Параметры протокола содержат параметры группы $G = \mathbb{F}_p^*$: характеристику поля, порядок группы и образующую:

ElGamalSignParameters ::= SEQUENCE {

prime INTEGER, – число $p$

$r$ INTEGER, – порядок группы $r$

generator INTEGER – образующая $a$

}

Параметры файла (сообщения) не используются, следует использовать последовательность нулевой длины. Дополнение открытого текста перед вычислением хэш-образа не производится.

*Приложение Г*

## ASN.1-ФОРМАТ ФАЙЛА ПОДПИСИ ГОСТ Р 34.10–2018

Общий формат файла подписи:

SEQUENCE {– заголовок

SET {– множество ключей, 1 задействован

SEQUENCE {– первый «ключ»

OCTET STRING – идентификатор алгоритма (протокол подписи ГОСТ)

80 06 07 00

UTF8String 'gostSignKey' – может не задействоваться

SEQUENCE {– значение открытого ключа

INTEGER - $x$ -координата точки $Q$

QX QX … QX

INTEGER - $y$ -координата точки $Q$

QY QY … QY

}

SEQUENCE {– параметры криптосистемы

SEQUENCE {– параметры поля

INTEGER – простое число $p$

PP PP … PP

}

SEQUENCE {– параметры кривой

INTEGER – коэффициент $A$ уравнения кривой

AA AA … AA

INTEGER – коэффициент $B$ уравнения кривой

BB BB … BB

}

SEQUENCE {– образующая группы точек кривой

INTEGER – $x$ -координата образующей точки $P$

PX PX … PX

INTEGER – $y$ -координата образующей точки $P$

PY PY … PY

}

INTEGER – порядок группы $q$
RR RR … RR
} – параметры криптосистемы
SEQUENCE {– подпись сообщения
INTEGER – число $r$
XR XR … XR
INTEGER – число $s$
SS SS … SS
}
}
SEQUENCE {} – параметры файла, не используются
}

**Все значения приведены в десятичной системе счисления**

## Вариант 1

```
p =  57896044620753384133869303843568937902752767818974600847634902975134129543643
r =  28948022310376692066934651921784468951377218528270520403696863131129758387393

a =  1
b =  52259530098387149819562511889780651425271270942919542722038553712464420235875
xP =  14539175448068301073584752148116082765715462525899666138074034449285211025933
yP =  83288014666338982823110297985564177671414910550363993483463248044786194004 51
```

## Вариант 2

```
p = 57896044628890729911196718984933305846544100325488685311213142875135838763683
r = 28948022314445364955598359492466652923270809441897180344196391207096541510137

a = 1
b = 51597193811365919768190236681066502033803499635094541650610225403695076439048
xP = 21371456824977467041033238171905463424508399897529674896678501178686263573482
yP = 52962982709744467108853563358242537068648343861092009194618855518747612108192
```

## Вариант 3

```
p = 57896044625414088412406986721186632159605151965036429316594800028484330862739
r = 28948022312707044206203493360593316079803694388568974763893400879284219004579

a = -1
b = 53520245325288251180656443226770638951803337703360722011463033447827147086694
xP = 36066034950041118412594006918367965339490267219250288222432003968962962331642
yP = 54906983586985298119491343295734802658016371303757622466870297979342757624191
```

## Вариант 4

```
p = 57896044625259982827082014024491516445703215213774687456785671200359045162371
r = 28948022312629991413541007012245758222850495633896873081323396140811733708403

a = -1
b = 53956679838042162451108292176931772631109916272820066466458395232513766926866
xP = 12933162268009944794066590054824622037560826430730236852169234350278155715869
yP = 18786030474197088418858017573086015439132081546303111294023901101650919011383
```

## Вариант 5

```
p =  57896044630612021680684936114742422271145183870487080309667128995208157569947
r =  28948022315306010840342468057371211135571302038761442251594012761075345324491

a =  1
b =  19750513962881385028059495396984460236743646692126413053976069443380491067343
xP =  43490682822985073571091311123441225129011272278165566160439297012894969619553
yP =  53273700124912449490307054424387372532482586733448415163119878489682918137700
```

## Вариант 6

```
p = 57896044622894643241131754937450315750132642216230685504884320870273678881443
r = 28948022311447321620565877468725157875067316353637126186229732812867492750347

a = 1
b = 41431894589448105498289586872587560387979247722721848579560344157562082667257
xP = 54672615043105947691210796380713598019547553171137275980323095812145568854782
yP = 42098178416750523198643432544018510845496542305814546233883323764837032783338
```

## Вариант 7

```
p = 57896044622894643241131754937450315750132642216230685504884320870273678881443
r = 28948022311447321620565877468725157875067316353637126186229732812867492750347

a = 1
b = 41431894589448105498289586872587560387979247722721848579560344157562082667257
xP = 54672615043105947691210796380713598019547553171137275980323095812145568854782
yP = 42098178416750523198643432544018510845496542305814546233883323764837032783338
```

## Вариант 8

```
p =  57896044628958718631213028275518411328476149599789770738757218840632915517411
r =  28948022314479359315606514137759205664236832023231628035871193493020981068937

a =  -1
b =  44516423948019661825420813927965592341675839270849860441861020678502941837466
xP =  23235760586019566647209667080457263089166278247417077298367088875172326850 58
yP =  20772302011053991390127435262297715010367018383131467831609444907978987653753
```

## Вариант 9

```
p = 57896044623304043104299010705220227593411047292536827792690781397890347747339
r = 28948022311652021552149505352610113796704505182029833954354922263763927570003

a = -1
b = 43710617685176640081115791799072828565464240578102647674571004997445073590733
xP = 23758980230668189222598764688099579588532528184549479557352832549279103197 97
yP= 41494802599112005706154126874294477637447794334726831230592311725680992538390
```

## Вариант 10

```
p = 57896044623332830704464930175758866532580959320654190378215212919875855638347
r = 28948022311666415352232465087879433266289459622068172692541485718987902249339

a = 1
b = 35800948915040763394854698676083084630967836106516964913271803528707388006 22
xP = 34716160583222611645030789937385729759908872067425542316787922148347820731789
yP = 58988212064147591382013637178025684997934448166901693688396851241678354665 30
```