## *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 с открытым ключом RSA является составной частью многих стандартов, а также используется во многих приложениях и протоколах, например, PGP, SSL/TLS, SSH и др.

Принцип действия криптосистемы RSA заключается в следующем. Пусть *p* и *q* – два больших простых числа и пусть . Выбирается число *e*, , взаимно простое со значением функции Эйлера , и вычисляется *d*  – мультипликативно обратное к *e*: , то есть . Числа *e* и *d* называются открытым и закрытым показателями соответственно. Открытым ключом является пара , закрытым ключом – число . Множители *p* и *q* должны храниться в секрете.

Для зашифрования сообщения *m* нужно вычислить шифртекст , а для расшифрования – возвести шифртекст в степень : .

Аналогично реализуется цифровая подпись: подписью сообщения *m* является число , где *d* – закрытый ключ формирования подписи.

1. **Применение криптосистемы RSA для шифрования**

Одна операция зашифрования в системе RSA позволяет зашифровать сообщение, представленное числом *m*, где 0 ≤ *m* < *n*. Текст более длинных сообщений необходимо разбивать на блоки. То же справедливо и для файлов: преобразование из потока байтов в число для операции шифрования 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 возможны 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).

Размер представляемого блока данных (последовательности байтов) должен быть на единицу меньше, чем длина числа *n* в байтах. Возможно также для хранения и передачи данных частично использовать старшие биты числа ; при этом необходимо осуществить побитовое разбиение открытого текста на блоки, что требует сложной реализации.

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*, размер блока – , длина ключа – . Для передачи каждого нового сообщения  генерируется случайный ключ симметричного шифрования *k*. Сгенерированный ключ представляется в виде числа и шифруется алгоритмом RSA. Основное сообщение разбивается на блоки длинной  и шифруется на ключе *k* симметричным шифром *C*.

Получателю такого составного сообщения передаются зашифрованный при помощи RSA ключ симметричного алгоритма  и основное сообщение . Получатель с помощью закрытого ключа *d* расшифровывает ключ симметричного алгоритма *k*, а затем и основное сообщение *m*.

Размер ключа  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 тело сообщения, передается зашифрованным на различных ключах зашифрования: , , …,. Получатели такого сообщения, каждый на своем закрытом ключе, расшифровывают промежуточный ключ *k* и получают доступ к телу сообщения.

Само тело сообщения присутствует в зашифрованном сообщении (файле) однократно. 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.

1. **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. Однако с развитием информатизации они стали терять уникальные физические качества, а печать теперь легко можно переснять сканером и воспроизвести в любом масштабе на цветном принтере.

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. Однако 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.

Схемы подписи, как и схемы шифрования с открытым ключом, основаны на вычислениях в конечной группе, поэтому их можно разделить по типу вычислимости порядка группы.

1. **Схема подписи RSA**

**Протокол 1.1.** Схема подписи RSA

*Вход отправителя*. Составное число , закрытый показатель .

*Вход получателя*. Составное число , открытый показатель .

*Результат*. Формирование и проверка подписи.

Для формирования подписи для сообщения  отправитель выполняет следующие действия.

1. Вычисляет хэш-образ  сообщения .
2. Зашифровывает число *r* на своем закрытом ключе , то есть вычисляет экспоненту .

Подписью для сообщения  является число .

Для проверки подписи получатель выполняет следующие действия.

1. Расшифровывает подпись  на открытом ключе  отправителя, то есть вычисляет , восстанавливая предполагаемый хэш-образ  сообщения .
2. Вычисляет хэш-образ  сообщения .
3. Проверяет равенство . Если оно выполняется, то результат: подпись подлинная, иначе результат: подпись неверна. ◼

Схема подписи RSA может быть рассмотрена как категория, объектами которой являются пары сообщение/подпись, а морфизмами – отображения таких пар, удовлетворяющие проверочным соотношениям. Очевидно, что если  – подлинное подписанное сообщение, то пара  тоже будет правильно подписанным сообщением, если  и  для некоторого целого . Новая пара  может быть создана следующим алгоритмом: выбрать сообщение , вычислить , найти логарифм  по основанию  и вычислить . Можно наоборот, сначала найти  для известного , а затем подобрать  так, чтобы выполнялось равенство . Поэтому стойкость подписи RSA не может превышать сложности вычисления логарифма в группе  или сложности обращения хэш-функции.

Если нарушитель составляет текст для подписи, то он может подготовить коллизию  такую, что, дать на подпись сообщение , а затем подменить подписанное сообщение  сообщением . В этом случае стойкость схемы подписи не может превышать сложности вычисления коллизий хэш-функции. Приведенные рассуждения показывают, что необходимо согласовывать сложности разложения, логарифмирования и нахождения коллизий.

1. **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. Представление ASN.1 предназначено для хранения и передачи структурированных данных. Подробное описание типов данных в ASN.1, форматов зашифрованного файла и файла подписи представлено в соответствии с *Приложением А*.

### Контрольные вопросы

1. Какие задачи положены в основу безопасности системы RSA?
2. Показать, что схема RSA работает корректно для любого сообщения .
3. Доказать, что задача разложения числа  на множители и задача вычисления функции Эйлера  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:

1. выполняет зашифрование и расшифрование файла с использованием алгоритма RSA. 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. 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;
2. выполняет формирование и проверку электронной подписи с использованием алгоритма RSA. При реализации алгоритма электронной подписи лучше всего использовать хэш-алгоритм SHA-256. Поскольку хэш-образ, вычисленный по алгоритму SHA-256, может быть длиннее модуля RSA, при проверке подписи следует сравнивать  с подписью сообщения *s*. Результатом данной части работы должна быть программа, предоставляющая пользователю возможность создать и проверить электронную подпись.

### Содержание отчета

1. Формулировка задания.
2. Выполненная работа:
3. использованные параметры криптосистемы;
4. сгенерированный ключ для симметричного алгоритма;
5. содержание зашифрованного файла в шестнадцатеричном виде (при значительном размере файла – первые 80-100 байт);
6. файл с подписью RSA.
7. Ответы на контрольные вопросы.
8. Выводы по работе. В выводах сделать предположения о возможных уязвимостях созданного программного обеспечения и о путях их устранения.
9. Листинг программы.