 Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ГЕНЕРАЦИИ

И ВЕРИФИКАЦИИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ

ПОДПИСИ

Студент: Коктыш Е. с.

ФИТ 3 курс 6 группа

Преподаватель: Нистюк О. А.

Минск 2024

1. **Теоретические сведения**

Электронная цифровая подпись (ЭЦП) является важным элементом современных информационных систем, использующих методы и технологии криптографического преобразования информации.

Понятие «электронная цифровая подпись» было введено в 1976 году У. Диффи и М. Хеллманом.

После создания RSA разработаны алгоритмы цифровой подписи И. Рабина и Р. Меркле. В 1984 году Ш. Гольдвассер, С. Микали и Р. Ривест сформулировали требования безопасности к алгоритмам ЭЦП, описали атаки на ЭЦП.

Определение 1. Электронная цифровая подпись – контрольная характеристика сообщения, которая вырабатывается с использованием личного ключа, проверяется с использованием открытого ключа, служит для контроля целостности и подлинности сообщения и обеспечивает невозможность отказа от авторства.

Таким образом, ЭЦП выполняет те же функции, что и собственноручная (поставленная «от руки») подпись:

* аутентифицировать лицо, подписавшее сообщение;
* контролировать целостность подписанного сообщения;
* защищать сообщение от подделок;
* доказать авторство лица, подписавшего сообщение, если это лицо отрицает свое авторство.

Важнейшими отличительными особенностям ЭЦП являются:

* ЭЦП представляет собой бинарную последовательность (в отличие от графического образа, каковым является подпись от руки);
* указанная бинарная последовательность зависит от содержания подписываемого сообщения.

Как следует из определения 1, основным компонентом в технологии ЭЦП является ключ. Принадлежность ключа, в предположении, что он известен только законным пользователям, позволяет решать все «возложенные на ЭЦП», сформированную на основе этого ключа, задачи. В соответствии с этим обстоятельством перечисленные выше функции ЭЦП могут быть реализованы на основе классических методов зашифрования/расшифрования:

* на основе симметричных систем (с тайным ключом),
* на основе симметричных систем и посредника,
* на основе асимметричных систем (с открытым ключом).

Первый из перечисленных методом ничем не отличается, например, от DES.

Во втором случае создаются по существу две симметричные системы: между отправителем и посредником и между посредником и получателем. Причем посредник выдает двум сторонам различный тайный (для иных субъектов системы) ключ.

Во последнем случае сообщение, отправляемое получателю, шифруется тайным ключом отправителя. Отправитель же верифицирует подпись (в данном случае – устанавливает авторство, используя для расшифрования публичный ключ отправителя, и получает гарантию в защищенности переданного сообщения от подделок, если после расшифрования формат и содержание документа имеют логическую стройность) с помощью открытого ключа отправителя.

Таким образом, в этом случае, как и в первых двух случаях, ЭЦП, как отдельный, самостоятельный, присоединенный к исходному документу элемент получаемого сообщения, отсутствует. Кроме, того, в отличие от классической асимметричной криптографии, где используется ключевая информация получателя, в нашем случае используется ключевая информация отправителя: открытый ключ – для зашифрования, тайный – для расшифрования.

С учетом изложенного можем сформулировать определение ЭЦП в несколько ином виде.

Определение 2. Электронная цифровая подпись – бинарная (или в ином виде) последовательность символов, являющаяся реквизитом электронного документа, зависящая от содержания этого документа и предназначенная для подтверждения целостности и подлинности электронного документа.

**ЭЦП на основе алгоритма RSA:**

Здесь можно рассматривать две ситуации:

* сообщение М0 подписывается и передается в открытом (незашифрованном) виде;
* сообщение М0 подписывается и передается в зашифрованном.

При этом подпись S вычисляется на основе известного из лабораторной работы №8 соотношения:

S = (H(Mo))^d0 mod n0 (1.1)

при указанном выше реверсе в отношении ключевой информации; в (1.1) d0 и n0 – элементы тайного ключа отправителя. Передаваемое сообщение М' = М0||S.

Если подписываемое сообщение М (М') также должно передаваться в зашифрованном виде, то обычно М' шифруется на стороне отправителя стандартным образом: с помощью открытого ключа получателя (еn и nn), который перед основным процессом верификации подписи расшифровывает послание своим тайным ключом: dn и nn . Далее осуществляются вычисления и анализ, как и в первом случае.

**ЭЦП на основе алгоритма Эль-Гамаля:**

Ключевая информация отправителя для ЭЦП создается точно так же, как это описано в материалах к ЛР №8. Она состоит из тех же элементов, что и ключи в DSA. Основное отличие в применении расчетов состоит в том, что результатом зашифрования является только одна пара чисел, а не пара – для каждого блока исходного сообщения. Причем в рассматриваемом случае таким сообщением является хеш подписываемого документа: Н(M0).

Итак, ключевая информация отправителя: открытый ключ: y, g и р; тайный ключ: х. Чтобы подписать сообщение Мо, обладатель используемых для ЭЦП ключей должен выбрать, как и в предыдущей схеме, случайное число k, взаимно простое с р-1.

Получателю отправляется сообщение М' = М0||S.

Для верификации подписи вычисляется хеш полученного сообщения, Н(Мn) = h. Далее нужно убедиться, что выполняется равенство:

y^a \* a^b = g^h (mod p) (1.5)

Если равенство выполняется, подпись верифицируется.

**ЭЦП на основе алгоритма Шнорра:**

Рассматриваемая схема является основой стандарта ЭЦП в Беларуси. Алгоритм ЭЦП К. Шнорра (К. Schnorr) является вариантом алгоритма ЭЦП Эль-Гамаля.

Одной из особенностей ЭЦП Эль-Гамаля является то, что число p должно быть очень большим, чтобы сделать действительно трудной проблему дискретного логарифма. Рекомендуется длина p, по крайней мере, должна составлять 1024 бита. Чтобы уменьшить размер подписи, Шнорр предложил новую схему, но с уменьшенным размером подписи.

1. **Практическая часть**

В данной лабораторной работе необходимо разработать пользовательское приложение, которое должно реализовывать генерацию и верификацию ЭЦП на основе алгоритмов RSA, Эль-Гамаля и Шнорра.

**2.1. RSA**

Для генерации и верификации ЭЦП на основе алгоритма RSA была написана функция, которая хеширует введенный текст при помощи алгоритма MD-5, после чего полученный хеш зашифровывается алгоритмом RSA.

Алгоритм генерации подписи заключается в следующих операциях:

̶ выбор простых чисел *p*, *q*;

̶ выбор случайного числа *e*, взаимно простого с функцией Эйлера *ф*(*n*)=(*p*-1)\*(*q*-1);

̶ нахождение числа *d*, такого что *ed* = 1\*mod (*p*-1)\*(*q*-1);

̶ вычисление хеш-образа сообщения *h*=*H*(*M*);

̶ вычисление ЭЦП: *S* = *hd* \*mod *n*.

Далее полученный открытый ключ {*e*, *n*} и письмо с ЭЦП {*m*, sign} будут отправлены получателю.

Для того, чтобы получатель смог провести операцию верификации сообщения, необходимо выполнить следующие действия:

̶ вычислить *h* = *Se*\*mod *n*;

̶ сравнить значение выше с полученным *h*=*H*(*M*);

Если полученные значения совпали, подпись верифицирована.

Алгоритм шифрования представлен в листинг 2.1.

|  |
| --- |
| ServerSignRSA {      constructor() {          const {privateKey, publicKey} = crypto.generateKeyPairSync('rsa', {              modulusLength: 2048,              publicKeyEncoding: {type: 'pkcs1', format: 'pem'},              privateKeyEncoding: {type: 'pkcs1', format: 'pem'}          });          prKey = privateKey;          pubKey = publicKey;          s = crypto.createSign('SHA256');      }      getSignContext(rs, cb) {          rs.pipe(s);          rs.on('end', () => {              cb({                  signature: s.sign(prKey).toString('hex'),                  publicKey: pubKey.toString('hex')              });          });      }; |

Листинг 2.1 – Код

На рисунке 2.1 представлен результат генерации и верификации электронной цифровой подписи на основе алгоритма RSA, а также время выполнения генерации ЭЦП.

|  |
| --- |
|  |

Рисунок 2.1 – Верификация электронной цифровой подписи на основе алгоритма RSA

В данном случае выбранный ключ, возможно, имеет большую длину или использует сложные математические операции, что приводит к формированию подписи с достаточно большим размером.

Длинная подпись может быть полезной в контексте криптографии, так как она обеспечивает более высокий уровень безопасности и устойчивости к атакам. Однако, следует также учитывать, что более длинная подпись может потребовать больше вычислительных ресурсов и времени для ее генерации и проверки. При выборе длины подписи необходимо учитывать баланс между безопасностью и эффективностью системы.

**2.2. Эль-Гамаля**

Алгоритм генерации подписи заключается в следующих операциях:

̶ выбор простого числа *p*;

̶ вычисление *g*, причем *g*<*p* – первообразный корень по модулю *p*;

̶ выбор *x*, меньшего *p*;

̶ вычисление *y* = *gx* mod *p*.

В результате зашифрования сообщения с ЭЦП на выходе будет лишь одна пара чисел, не для каждого блока сообщения.

Далее необходимо проделать следующие операции:

̶ выбрать *k* – взаимно простое число с (*p*-1);

̶ вычислить *a* = *gk* mod *p*;

̶ вычислить *b* = k-1 (*H*(*Mo*) – *xa*) mod (*p*-1);

Пара чисел *S* = {*a*,*b*} и будет являться цифровой подписью. Далее получателю будет отправлено сообщение *M*’=*Mo*||*S*, которое является конкатенацией исходного сообщения и ЭЦП.

Для верификации подлинности полученного сообщения необходимо проверить равенство *yaab* (mod *p*) = *gh* (mod *p*), в которое подставляются все вычисленные ранее значения, *h*=*H*(*M*п) – хеш-образ полученного сообщения.

Если данное равенство выполняется, подпись верифицированна и подлинна.

Алгоритм шифрования представлен в листинг 2.2.

|  |
| --- |
| ServerSignElgam{      constructor(){          const {privateKey, publicKey} = crypto.generateKeyPairSync('dsa', {              modulusLength: 2048,              publicKeyEncoding: {type: 'spki', format: 'pem'},              privateKeyEncoding: {type: 'pkcs8', format: 'pem'}          });          prKey = privateKey;          pubKey = publicKey;          s = crypto.createSign('SHA256');      }      getSignContext(rs,cb){          rs.pipe(s);          rs.on('end', () => {              cb({                  signature: s.sign(prKey).toString('hex'),                  publicKey: pubKey.toString('hex')              });          });}; |

Листинг 2.2 – Результат работы алгоритма

На рисунке 2.2 результат генерации ЭЦП на основе алгоритма Эль-Гамаля, проверка подписи и время выполнения генерации.

|  |
| --- |
|  |

Рисунок 2.2 – Результат работы алгоритма Эль-Гамаля

В итоге получаем зашифрованное сообщение (последовательность чисел). ЭЦП в данном случае служит хеш, полученный на предыдущем этапе.

Для верификации сообщения необходимо расшифровать полученный шифротекст. В результате должен получится хеш исходного сообщения. Если они равны – сообщение не изменялось. Так же, электронная цифровая подпись на основе алгоритма Эль-Гамаля имеет достаточно небольшую длину, которая зависит от выбранных исходных ключей и параметров алгоритма. Это обеспечивает дополнительный уровень защиты и безопасности при использовании данной системы электронной цифровой подписи.

**2.3. Алгоритм Шнорра**

Одной из особенностей ЭЦП Эль-Гамаля является то, что число p должно быть очень большим, чтобы сделать действительно трудной проблему дискретного логарифма. Рекомендуется длина p, по крайней мере, должна составлять 1024 бита. Чтобы уменьшить размер подписи, Шнорр предложил новую схему, но с уменьшенным размером подписи.

Алгоритм генерации ключевой информации заключается в следующих шагах:

̶ генерация простых чисел *p*, *q*;

̶ вычисление (*p*-1) – делителя;

̶ выбор любого *g*≠1, такого что *gq* = 1 mod *p*;

̶ выбор любого числа *х*<*q* – это и будет закрытый ключ;

̶ вычисление *y* = *g–X* mod *p*;

Для верификации сообщения необходимо реализовать следующий алгоритм:

̶ выбор случайного числа *k* < *q*;

̶ вычислить *a* = *gk* mod *p*;

̶ создать хеш-образ сообщения: *h* = *H*(*Mo*||*a*);

̶ вычислить *b* = (*k*+*xh*) mod *q*;

Получателю будет отправлено сообщение *M*’ = *Mo*||*S*, являющееся конкатенацией исходного сообщения и ЭЦП *S*={*h*,*b*}.

Для проверки подписи на подлинность необходимо вычислить *X* = *gbyh*(mod *p*), после чего проверить выполняется ли равенство вычисленного ранее *h* хеш-образа и *H*(*М*п||*X*) хеш-образа конкатенации полученного сообщения с вычисленным значением *Х*. Если равенство выполняется, подпись верифицирована.

Алгоритм шифрования представлен в листинг 2.3.

|  |
| --- |
| class ServerSignShnorr{      constructor(){          //It's would be ok on nodejs v12.0.0          const {privateKey, publicKey} = crypto.generateKeyPairSync('ed25519', {              modulusLength: 2048,              publicKeyEncoding: {type: 'spki', format: 'pem'},              privateKeyEncoding: {type: 'pkcs8', format: 'pem'}          });          prKey = privateKey;          pubKey = publicKey;      }      getSignContext(rs){          return {              signature: crypto.sign(null, Buffer.from(rs), prKey),              publicKey: pubKey.toString('hex')          };      };} |

Листинг 2.3 – Результат работы алгоритма

На рисунке 2.3 же представлена вышеперечисленная информация, отличие лишь в алгоритме. Здесь используется алгоритм Шнорра.

|  |
| --- |
|  |

Рисунок 2.3 – Результат работы алгоритма Шнорра

На основе выбранного ключа была создана подпись, которая имеет не значительную длину. Это объясняется тем, что используемый ключ имеет определенные параметры и свойства, которые влияют на конечную длину подписи. В данном случае выбранный ключ, возможно, имеет не большую длину или использует сложные математические операции, что приводит к формированию подписи с достаточно не большим размером.

**3.Время генерации и верификации ЭЦП**

Для оценки времени генерации был построен график времени выполнения генерации ЭЦП при помощи разных алгоритмов. Графики представлен на рисунке 3.1. и 3.2

|  |
| --- |
|  |

Рисунок 3.1. – Графики времени выполнения генерации

График представлен на рисунке 3.2.

|  |
| --- |
|  |

Рисунок 3.2. – Графики зависимости времени

Из представленных графиков видно, что время, необходимое для создания и проверки электронной подписи, остается практически неизменным для всех типов алгоритмов и независимо от размера исходного сообщения. Тем не менее, наблюдаются незначительные различия в скорости вычислений между алгоритмами. Так, алгоритм RSA обычно требует чуть больше времени по сравнению с алгоритмами Эль-Гамаля и Шнорра, хотя эти различия и не являются значимыми.

**Вывод**

В ходе выполнения данной лабораторной работы мы изучили и приобрели практические навыки генерации и верификации электронной цифровой подписи. Так же был построен график выполнения генерации ЭЦП при помощи алгоритмов RSA, Эль-Гамаля и Шнорра.