

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И
РАДИОЭЛЕКТРОНИКИ

Кафедра радиотехнических систем

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторной работе

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

по курсу

ЗАЩИТА ИНФОРМАЦИИ

для студентов специальности 39 01 02 «Радиоэлектронные системы»

Минск 2002

УДК 681. 327 (075.8)

ББК 32.811 я 7

М 34

Составитель С. Б. Саломатин

Методические указания к лабораторной работе КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ по курсу «Защита информации» для студентов специальности 39 01 02 «Радиоэлектронные системы»/ Сост. С.Б. Саломатин. - Мн.: БГУИР, 2002.- 22 с.

Методические указания содержат теоретические сведения, алгоритмы, программы моделирования криптографических протоколов, использующих метод доказательства с нулевым разглашением знаний. В лабораторной работе исследуются статистические характеристики атак на криптографические протоколы. Моделируется и исследуется пакетная радиосеть, использующая протоколы аутентификации субъектов.

УДК 681. 327 (075.8)

ББК 32.811 я 7

© С. Б. Саломатин,
составление, 2002

© БГУИР, 2002

1. ЦЕЛЬ РАБОТЫ

1. Изучить методы защиты информации с помощью криптографических протоколов, использующих алгоритм доказательства с нулевым разглашением знаний.
2. Исследовать алгоритмы анализа и синтеза криптографических протоколов.
3. Получить навыки программирования криптографических протоколов.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Идентификация и аутентификация

Защита информации в радиоэлектронных системах предполагает решение задачи идентификации и аутентификации. Идентификация устанавливает взаимно однозначное соответствие между множеством сущностей системы и множеством идентификаторов. Аутентификация представляет собой проверку подлинности идентификаторов.

В процессе идентификации и аутентификации пользователь или программа (претендент) запрашивает доступ у системы (верификатора). Сначала осуществляется идентификация. Верификатор требует от претендента предъявить некоторый идентификатор и проверяет принадлежность идентификатора I множеству зарегистрированных в системе. В случае корректности идентификатора верификатор выполняет процедуру аутентификации (например, запрашивает некоторые данные) чтобы убедиться, что претендент является именно тем, за кого себя выдает. Доступ претендента в систему разрешается только в случае успешного завершения процедуры аутентификации. Часто устанавливается некоторое пороговое значение для числа попыток предъявления некорректного идентификатора, при превышении которого все дальнейшие попытки данного претендента к системе блокируются.

Различают два основных типа аутентификации субъекта и объекта. Процедуры *аутентификации субъекта* подразумевают обмен информацией, имеющей своей целью убедить верификатора в подлинности идентификатора, предъявляемого претендентом. *Аутентификация объекта* предполагает проверку подлинности идентификатора, представленного в совокупности с некоторыми данными. Отличие от аутентификации субъекта состоит в том, что в этом случае претенденту не нужно быть активным участником процесса аутентификации.

2.2. Метод доказательства с нулевым разглашением знаний

Для аутентификации субъекта может применяться метод доказательства с нулевым разглашением знаний, суть которого состоит в том, что наличие информации у претендента проверяется без раскрытия этой информации (или ее части)

верификатору или третьей стороне. Как правило, используется интерактивный режим работы. Верификатор задает претенденту ряд вопросов. Претендент вычисляет ответ на каждый вопрос с помощью имеющейся у него секретной информации. По этим ответам верификатор может с достаточно высокой степенью уверенности установить, что претендент действительно обладает секретной информацией (хотя никакая часть этой информации в ответах фактически не раскрывается).

Рассмотрим следующую *модель протокола аутентификации*. В распоряжении претендента и верификатора имеются вероятностные машины Тьюринга P и V соответственно. Вычислительные ресурсы, которые может использовать претендент, неограничены, в то время как машина V работает за полиномиальное время. Машины P и V имеют общую коммуникационную ленту для обмена сообщениями. После записи сообщения на коммуникационную ленту машина переходит в состояние ожидания и выходит из него, как только на ленту будет записано ответное сообщение. Машины P и V имеют также общую входную ленту, на которую записано входное слово x . Утверждение, которое доказывает претендент, суть " $x \in L$ ", где L – некоторый фиксированный (известный претенденту и верификатору) язык. Чтобы избежать тривиальности, язык L должен быть трудным (например, NP-полным), иначе верификатор может самостоятельно проверить, что $x \in L$. Протокол доказательства состоит в том, что верификатор, используя случайность, выбирает некоторые вопросы, задает их претенденту и проверяет правильность ответов. Выполнение протокола завершается когда машина V останавливается, при этом она выдает 1 "YES", если доказательство принято, и 0 "NO" – в противном случае.

Определение. Интерактивный протокол доказательства для языка L называется доказательством с абсолютно нулевым разглашением, если выполняются следующие условия.

1. *Полнота.* Для всех $x \in L$ вероятность

$$\Pr\{[P(x), V(x)] = 1\} = 1,$$

где $[P(x), V(x)]$ – выходное слово (случайная величина) машины P , когда P и V работают на входном слове x .

Полнота означает, что если входное слово принадлежит языку L и оба участника (претендент и верификатор) следуют протоколу, то доказательство будет всегда принято.

2. *Корректность.* Для любой машины Тьюринга P^* , для любого полинома p и для всех $x \notin L$ достаточно большой длины

$$\Pr\{[P^*(x), V(x)] = 1\} < \frac{1}{p(|x|)},$$

где $|x|$ – длина слова x .

Требование корректности защищает верификатора от нечестного претендента, который пытается обмануть его, “доказывая” ложное утверждение. При этом претендент может каким угодно образом отклоняться от действий, предписанных протоколом, т. е. вместо машины Тьюринга P использовать любую другую машину P^* . Требуется, чтобы вероятность обмана была в любом случае пренебрежительно мала.

3. *Свойство нулевого разглашения.* Для любой полиномиальной вероятностной машины Тьюринга V^* существует вероятностная машина Тьюринга M_{V^*} , работающая за полиномиальное в среднем время, и такая, что для всех $x \in L$ выходное слово машины M_{V^*} удовлетворяет равенству

$$M_{V^*}(x) = [P(x), V^*(x)].$$

Машина M_{V^*} называется моделирующей машиной для V^* . Предполагается, что математическое ожидание времени ее работы ограничено полиномом от длины x . Машина M_{V^*} в принципе может работать достаточно долго, но вероятность того, что время ее работы превысит некоторую полиномиальную границу, мала. Свойство нулевого разглашения защищает претендента от нечестного верификатора, который, произвольно отклоняясь от действий, предписанных протоколом (используя V^* вместо V) пытается извлечь из его выполнения дополнительную информацию. Верификатор может при этом получить только такую информацию, которую он смог бы вычислить и самостоятельно (без выполнения протокола) за полиномиальное время.

Если в определении свойства нулевого разглашения заменить равенство случайных величин требованием, чтобы их распределения вероятностей «почти не отличались», то получим *доказательства со статистическим нулевым разглашением*. Существует еще один тип доказательств – *доказательства с вычислительно нулевым разглашением*. В этом случае требуется, чтобы моделирующая машина создавала распределение вероятностей, которое неотлично от $[P(x), V^*(x)]$ никаким полиномиальным вероятностным алгоритмом.

2.3. Криптографические протоколы аутентификации субъекта на основе метода доказательства с нулевым разглашением знаний

Протокол Фиата-Шамира (FS). В процессе аутентификации участвуют претендент, верификатор (проверяющий) и доверенная сторона (центр управления и распределения ключей). Выполняются t раундов проверки. Каждый раунд реализует трехшаговый протокол интерактивного обмена информацией и проверку знаний претендента.

Задание системных параметров. Центр управления и распределения ключей выбирает и публикует в сертифицированном справочнике число $n = p \cdot q$, где p и q – достаточно большие, простые числа. Значения множителей p и q в справочнике не публикуются и являются секретной информацией доверенной стороны. Каждый претендент A осуществляет следующие предварительные действия:

- 1) выбирает случайное, взаимно простое с n число s , $\text{НОД}(s, n) = 1$, $1 \leq s \leq n$;
- 2) вычисляет целое число $v = s^2 \bmod n$, являющееся квадратичным вычетом по модулю n , объявляет его открытым ключом и регистрирует его в справочнике центра управления и распределения ключей.

Секретным ключом претендента A является целое число s .

Протокол интерактивного обмена содержит три шага передачи и приема информации между претендентом и верификатором.

1. Претендент выбирает случайное число r , $1 \leq r \leq n - 1$, вычисляет $x = r^2 \bmod n$, и посылает число x верификатору B .
2. Верификатор B посылает претенденту A случайный бит $e = 0$ или $e = 1$.
3. Претендент вычисляет и посылает верификатору число $y = r$ (если $e = 0$) или число $y = r \cdot s \bmod n$ (если $e = 1$).

Проверка знаний претендента. На этом шаге проверяется соотношение

$$y^2 \equiv x \cdot v^e \bmod n.$$

Если $e = 0$, то верификатор проверяет, что

$$y^2 \equiv x \bmod n.$$

чтобы убедиться, что претендент знает квадратный корень из x .

Если $e = 1$, то верификатор проверяет, что

$$y^2 \equiv x \cdot v \bmod n.$$

чтобы убедиться, что претендент знает квадратный корень из $v^{-1} = s \bmod n$.

Если $y = 0$, то верификатор считает, что претендент не прошел проверку.

Стороны повторяют этот цикл t раз при разных случайных значениях e и r . Если претендент A не знает значения s , он может выбрать такое r , которое позволяет ему обмануть верификатора в случае $e = 0$ или $e = 1$, но не в обоих случаях одновременно. Вероятность успешной атаки на протокол на основе подмены в одном раунде составляет 0,5. Вероятность обмана в t циклах равна 2^{-t} .

Количество раундов протокола FS можно уменьшить, если использовать векторную форму представления информации.

Протокол Фейге-Фиат-Шамира (FFS). Предполагает участие трех действующих лиц: претендента, верификатора и центра управления и распределения ключей.

Формирование системных параметров. Центр управления и распределения ключей определяет общий для всех пользователей модуль $n = p \cdot q$, где p и q - простые числа, причем выполняются следующие условия

$$p \equiv 3 \pmod{4} \quad \text{и} \quad q \equiv 3 \pmod{4}.$$

В этом случае модуль n труднее поддается процессу факторизации, что является дополнительной защитой протокола. Кроме того, если n – положительное нечетное целое число и $n = n_1 n_2$ – его разложение на простые сомножители и если $\left(\frac{a}{n}\right)$ - символ Якоби

$$\left(\frac{a}{n}\right) = -1,$$

тогда число a является квадратичным невычетом по модулю n . Это свойство можно использовать для проверки правильности выбора открытого ключа претендента

После вычисления значения модуля, центр управления и распределения ключей определяет размер вектора k и число раундов проверки t . Числа k и t являются секретными параметрами системы аутентификации.

Формирование ключей претендентов. Каждый претендент A выполняет следующие действия:

- определяет вектор-строку случайных чисел $S = (s_1, s_2, \dots, s_k)$, каждое значение $\{s_i\}$ лежит в диапазоне $1 \leq s_i \leq (n - 1)$ и проверяет выполнение равенства $\text{НОД}(s_i, n) = 1$ (отметим, что взаимная простота чисел s_i и n в поддерживается правилом формирования модуля);
- определяет вектор-строку случайных бит $B = (b_1, \dots, b_k)$;
- вычисляет множество значений

$$v_i = (-1)^{b_i} (s_i^2)^{-1} \pmod{n}, \quad \text{для } 1 \leq i \leq k;$$

- устанавливает связь и проходит процесс идентификации и авторизации в центре управления и распределения ключей, после чего регистрирует в центре свой открытый ключ $V = (v_1, \dots, v_k)$;

- центр управления и распределения ключей проверяет корректность значений представленного ключа, вычисляя символ Якоби $J(v_i) \pmod{n}$. Если символ Якоби равен $+1$, то значения ключа корректны, в противном случае ключ не регистрируется.

Вектор s является секретным ключом претендента не подлежащим разглашению.

Протокол раунда информационного обмена. В каждом раунде претендент A и верификатор B обмениваются информацией по следующему алгоритму.

1. Претендент A вычисляет случайное число r , $1 \leq r \leq (n - 1)$. Выбирает случайным образом бит из вектора b , после чего вычисляет число

$$x = (-1)^{b_i} r^2 \pmod{n},$$

которое посылает верификатору B : $A \rightarrow B: x = \pm r^2 \pmod{n}$.

2. В ответ верификатор В генерирует случайным образом и посылает претенденту А вектор $\mathbf{e} = (e_1, \dots, e_k)$, содержащий k бит информации:

$$A \leftarrow B: \mathbf{e} = (e_1, \dots, e_k) \quad e_i \in \{0, 1\}.$$

3. Претендент А вычисляет и посылает обратно верификатору В число y

$$A \rightarrow B: y = r \prod_{j=1}^k s_j^{e_j} \bmod n.$$

Проверка знаний претендента. Верификатор В вычисляет число

$$z = y^2 \prod_{j=1}^k v_j^{e_j} \bmod n,$$

которое сравнивает с ранее принятым числом x .

Если

$$z = \pm x \text{ и } z \neq 0,$$

то на данном раунде подлинность представленного идентификатора подтверждается. Считается, что претендент доказал верификатору наличия у него требуемых знаний, для выбранного размера вектора k . Для полной успешной аутентификации претендент должен подтвердить подлинность представляемого идентификатора в заданном количестве раундов.

Вероятность успешной криптоатаки на основе подмены с использованием открытых текстов в t циклах равна 2^{-kt} . Заметим, что верификатор заинтересован в применении как можно большего количества раундов для уменьшения вероятности обмана. С другой стороны, претендент хочет как можно быстрее пройти процесс аутентификации. Поэтому значение t в протоколе FFS выбирается на основе компромисса. Условие доказательства с нулевым разглашением знаний и приемлемый уровень защищенности обеспечиваются, если $k = O(\log \log n)$ и $t = \Theta(\log n)$.

Протокол Гиллоу-Куискуотера (GQ). Протокол GQ использует меньшее количество раундов информационного обмена при заданной вероятности доказательств корректности представленного идентификатора. Предполагается, что каждому претенденту присвоен уникальный идентификационный номер I_A и известно как вычислить хэш-функцию $h_A(I_A)$.

Системные параметры. Центр управления и распределения ключей по методике алгоритма RSA вычисляет:

1) общий для всех пользователей модуль $n = p \cdot q$, где p и q – простые, достаточно большие, целые числа.

2) случайное число v , являющееся открытым ключом и удовлетворяющее условиям $\text{НОД}(v, \phi) = 1$ и $v \geq 3$, где $\phi = (p-1)(q-1)$ – функция Эйлера.

3) секретный ключ $s = v^{-1} \bmod n$.

4) секретное аккредитационное число $s_A = (h_A)^{-s} \bmod n$, которое через надежный канал связи передается претенденту А.

Протокол информационного обмена. Претендент А доказывает верификатору В свою идентичность за три шага.

1. Претендент А выбирает случайным образом секретное r , $1 \leq r \leq (n - 1)$ и вычисляет представляемое доказательство

$$x = r^v \bmod n.$$

Претендент передает верификатору В пару (I_A, x) .

2. Верификатор В выбирает случайным образом и передает претенденту А число e , $1 \leq e \leq v$.

3. Претендент А вычисляет и передает верификатору В числа

$$y = r s_A^e \bmod n.$$

Проверка знаний претендента. Верификатор В принимает y , формирует хэш-функцию h_A и вычисляет

$$z = h_A^e y^v \bmod n$$

Схема принятия решения сравнивает числа z и x . Если $z = x$ и $z \neq 0$, то принимается решение, что претендент А доказал свои знания и раунд аутентификации завершен успешно. В противном случае выносится решение об отсутствии у претендента требуемых знаний. В общем случае возможна t -раундовая процедура аутентификации. Окончательное решение об успешной аутентификации в этом случае выносится после проведения за заданное время требуемого количества успешных раундов аутентификации.

Уровень защищенности протокола GQ определяется величиной открытого ключа v . Вероятность успешной атаки на основе подмены равна v^{-t} . Если v – постоянная величина, то значение t должно расти быстрее чем $(\log \log n)$. Условие доказательства с нулевым разглашением знаний обеспечивается, если $(tv) = O((\log n)^c)$, где c – постоянная величина. С другой стороны, для обеспечения полиномиального уровня конфиденциальности необходимо выполнения условия $v^t = O((\log n)^c)$. Для обеспечения приемлемого уровня защищенности необходимо, чтобы v^{-t} была меньше чем обратная полиномиальная функция от $\log n$.

3. МОДЕЛИРОВАНИЕ КРИПТОГРАФИЧЕСКИХ ПРОТОКОЛОВ В СРЕДЕ ПРОГРАММНОГО ПАКЕТА MAPLE

Ниже приводятся программы, моделирующие криптографические протоколы, написанные с помощью программного пакета Maple.

Программа FS «Криптопротокол Фиата-Шамира»

Системные параметры

```

> with(numtheory): with(linalg):
p:=nextprime(rand()*rand());      # Вычисление случайного простого числа p #
                                   p := 4344471250977410178133
> q:=nextprime(rand()*rand());      # Вычисление случайного простого числа q #
                                   q := 14285410350391387596689
> n:=p*q;                          # Вычисление модуля n #
                                   n := 62062554575690515136701003788398842251001637
> s:=(rand()*rand()*rand()) mod n; # Вычисление претендентом секретного ключа s #
                                   s := 189274957802787037181466862693390625
> igcd(s,n); # Проверка свойства взаимной простоты чисел n и s #
                                   1
> v:=s&^2 mod n; # Вычисление претендентом открытого ключа v #
                                   v := 33198894580960510655898176898615177311921923
                                   Задание количества раундов информационного обмена
> t:=3:

```

Моделирование процесса информационного обмена между претендентом и
верификатором

```

> randombit:=rand(0..1): # Задание генератора случайных бит
> for l to t do # Цикл t раундов информационного обмена #
> msg[l]:=rand(): r[l]:=msg[l] mod n; #Формирование претендентом A случайного
                                   числа r#
> x[l]:=r[l]&^2 mod n;            # Шаг 1. Вычисление претендентом A и посылка
                                   верификатору B сообщения x#
> b[l]:=randombit():             #Формирование верификатором B и посылка
                                   претенденту A случайного бита b#
> y[l]:=r[l]*s^b[l] mod n od:    #Вычисление претендентом A и посылка
                                   верификатору B числа y, являющегося
                                   доказательством того, что претендент знает
                                   квадратный корень из x и обладает секретным
                                   ключом s#
> X:=seq(x[i],i=1..t); # Сохранение числовых значений информационного обмена#
X      :=      306261242564368288305124,      129335369518901748101904,
503730496384756190095225
> B:=seq(b[i],i=1..t);

```

$D := 1, 0, 1$

➤ $Y := \text{seq}(y[i], i=1..t);$

$Y := 46887890651348441717672662635335788944832131, 359632269852, 32590950915207217501290418605370294283160657$

Проверка верификатором В правильности представленных доказательств .

Вычисление доказательных числовых последовательностей Z и Z1

> for l to t do

> $z[l] := y[l] \&^2 \bmod n;$

> $z1[l] := x[l] * v^b[l] \bmod n;$ od;

> $Z := \text{seq}(z[i], i=1..t);$

$Z := 30503041975052550048501541671915581135568218, 129335369518901748101904, 2126578184114901773698215281806882540262059$

> $Z1 := \text{seq}(z1[i], i=1..t);$

$Z1 := 30503041975052550048501541671915581135568218, 129335369518901748101904, 2126578184114901773698215281806882540262059$

Принятие решения по аутентификации претендента: "YES" – субъект аутентифицирован положительно; "NO" – субъект аутентифицирован отрицательно.

> if $Z=Z1$ then 'YES'; else 'NO' fi;

YES

Контроль работоспособности программы аутентификации при действии ложного секретного ключа s1

> $s1 := \text{rand}();$

$s1 := 542136407374$

> for l to t do $\text{msg}[l] := \text{rand}(); r[l] := \text{msg}[l] \bmod n; x[l] := r[l] \&^2 \bmod n;$

> $b[l] := \text{randombit}();$

> $y[l] := r[l] * s^b[l] \bmod n$ od;

> $X := \text{seq}(x[i], i=1..t);$

$X := 11279630644507088103529, 666271838047429196361, 442609411406876723785801$

> $B := \text{seq}(b[i], i=1..t);$

$B := 0, 1, 1$

> $Y := \text{seq}(y[i], i=1..t);$

$Y := 106205605523, 13993756154448864248106, 360677371166281656205526$

> for l to t do

```

> z[l]:=y[l]&^2 mod n;
> z1[l]:= x[l]*v^b[l] mod n; od;
> Z:=seq(z[i],i=1..t);
Z := 11279630644507088103529, 762494351033750549967631218154036257141805,
2085212540172243810631916698491025978264564
> Z1:=seq(z1[i],i=1..t);
Z1 := 11279630644507088103529, 4093107792655774235724882838049641797688334,
1225529678146243167639872959089838482557204
> if Z=Z1 then `YES`; else `NO` fi;

```

NO

Программа FFS “Криптопротокол Фейге – Фиата - Шамира”

Системные параметры

```

> with(numtheory): with(linalg):
> p1:=0; for i to 5 while p1<>3 do p2:=nextprime(rand()); > p1:=p2 mod 4 od:
> if p1=3 mod 4 then p:=p2; fi; # Вычисление числа p #
p := 427419669163
> q1:=0: > for i to 10 while q1<>3 do q2:=nextprime(rand()); > q1:=q2 mod 4 od:
> if q1=3 mod 4 and q2<>p then q:=q2; else q:='FFF' fi; # Вычисление числа q #
q := 343633073743
> n:=p*q; # Вычисление модуля n #
n := 146875534692697842087109
> k:=5: # Задание секретного числа k #
> for i to k do s[i]:= rand(); nod:= igcd(s[i],n) od:
> S:=seq(s[l2],l2=1..k); # Вычисление вектора секретного ключа S #
S := 474256143563, 558458718976, 746753830538
> randombit:=rand(0..1):
> for i to k do b[i]:= randombit() od:
> B:=seq(b[l2],l2=1..k); # Вычисление вектора случайных бит B #
D := 1, 0, 1
> for i to k do v[i]:=((-1)&^b[i])*((s[i]&^2)&^(-1)) mod n; od:
> V:=seq(v[l2],l2=1..k); # Вычисление вектора открытого ключа V #
V := 140322027103855424571661, 89167569235036397477018,
110091725873937243124961
> for i to k do J(v[i],n) od; # Вычисление символа Якоби для элементов вектора V #

```

1

1

Моделирование процесса информационного обмена

```

> msg:=rand(n); r:=msg() mod n;# Формирование претендентом случайного числа r #
      r := 11202363945770601517110
> d1:=0: while d1=0 do d1:=(rand() mod k); od: > if d1<>0 then d:=d1 fi;
> x:=((-1)&^b[d])*(r&^2) mod n; # Вычисление претендентом числа x #
      x := 117297801318216640941019
> for i to k do e[i]:=randombit() od:
> EV:=seq(e[l2],l2=1..k); # Вычисление верификатором вектора случайных чисел EV #
      EV := 0, 1, 0
> y:=evala(r*product(s[l]&^e[l] mod n, l=1..k)) mod n;# Вычисление претендентом
      числа y #
      y := 47503653242287051071119

```

Проверка верификатором представленных доказательств

```

> g:=evala(product(v[l]&^e[l] mod n, l=1..k)) mod n;
> z:=(y&^2)*g mod n;
      z := 117297801318216640941919
> z1:=-z mod n;
      z1 := 29577733374481201145190
> if x=z or x=z1 and z<>0 then `YES`; else `No`; fi;
      YES

```

Проверка работоспособности программы при действии ложного секретного ключа s1

```

> for i to k do s1[i]:= rand(); nod:= igcd(s1[i],n) od: > seq(s1[l2],l2=1..k);
      453747019461, 644031395307, 920624947349
> y:=evala(r*product(s1[l]&^e[l] mod n, l=1..k)) mod n;
> g:=evala(product(v[l]&^e[l] mod n, l=1..k)) mod n;
> z:=(y&^2)*g mod n;
      z := 14290534614692728120783
> z1:=-z mod n;
      z1 := 132585000078005113966326
> if x=z or x=z1 and z<>0 then `YES`; else `No`; fi;
      NO

```

Программа GQ «Криптопротокол Гиллоу-Куискуотера»

Системные параметры

```
> with(numtheory): with(linalg):
> p:=nextprime(rand()*rand()); # Вычисление случайного простого числа p #
      p := 13558693922547573765631
> q:=nextprime(rand()*rand()); # Вычисление случайного простого числа q #
      q := 171675368879301488681177
> n:=p*q; # Вычисление модуля n #
      n := 2327693780674897974390356490196290213879227687
> f:=(p-1)*(q-1); # Вычисление функции Эйлера #
      f := 2327693780674897974390171256133488364816780880
> v1:=0: while igcd(v1,f)<>1 do v1:=rand() od:
> if igcd(v1,f)=1 then v:=v1 fi; # Вычисление открытого ключа v #
      v : 432613486829
> igcd(v,f); # Проверка условия взаимной простоты чисел v и f #
      1
> s:=v&^(-1) mod f; # Вычисление секретного ключа s #
      s := 1325273564898245661753935175617017240295093909
```

Выбор параметров пользователя

```
➤ h1:=0: while igcd(h1,f)<>1 do h1:=rand() od:
➤ if igcd(h1,f)=1 and h1<>v then h:=h1; else `return` fi; # Задание идентификацион-
      ного номера (хэш-функции) претендента h #
      h := 509974546787
> sa:=h&^(-s) mod n; # Секретный аккредитационный номер претендента sa #
      sa := 35747566095833209920165128744418582701237689
```

Моделирование протокола информационного обмена

```
> r1:=rand():
> if r1<(n-1) then r:=r1; fi; # Вычисление секретного числа пользователем A #
      r := 830196465066
> x:=r&^v mod n; # Вычисление числа x претендентом A #
      x := 1001221517916060202352373217934722793092929974
> e1:=rand(): while e1>v do e1:=rand() od:
> if e1<v then e:=e1 fi; # Вычисление случайного числа e верификатором B #
      e := 151776562728
```

```
> y:=r*sa&^e mod n; # Вычисление числа y претендентом A #
y := 335000620497233576938771090545310137234505100
```

Проверка верификатором представленных доказательств

```
> z:=(h&^e)*(y&^v) mod n;
z := 1001221517916060202352373217934722793092929974
> if z=x then `YES` else `NO` fi;
YES
```

Контроль работоспособности программы при действии ложного секретного ключа s2

```
> s2:= rand();
s2 := 166993783344
> sa2:=h&^(-s2) mod n;
712312218508352562559110564294280957253654467
> y:=r*sa2&^e mod n;
y := 1928529588033135094205474699836033259083683850
> z:=(h&^e)*(y&^v) mod n;
z := 1483848403250319532429895927813879720267535789
> if z=x then `YES` else `NO` fi;
NO
```

Программа “Tst1 FS”

Задание системных параметров

```
> with(numtheory): with(linalg):
> p:=nextprime(rand()*rand()*rand()); # Вычисление простого числа n #
p := 33110390636697557070951067268783269
> q:=nextprime(rand()*rand()*rand()); # Вычисление простого числа p #
q := 40577461120852497458034316960779877
> n:=p*q; # Вычисление модуля n #
n := 1343535588756833691097805930416239354759703345228672452179319129477913
> s:=(rand()*rand()*rand()) mod n; # Вычисление секретного ключа s #
s := 106669165247031581394326055711154547
> igcd(s,n); #Проверка на взаимную простоту секретного ключа и модуля #
1
> v:=s&^2 mod n; # Вычисление открытого ключа v #
v := 630026104443860564166746963756297689943095612384793915950406682951905
```

Задание параметров теста

```
> w:=100:           # число попыток подстановки ложного секретного ключа s2 #
> t:=10:            # количество t раундов информационного обмена #
```

Моделирование процесса информационного обмена

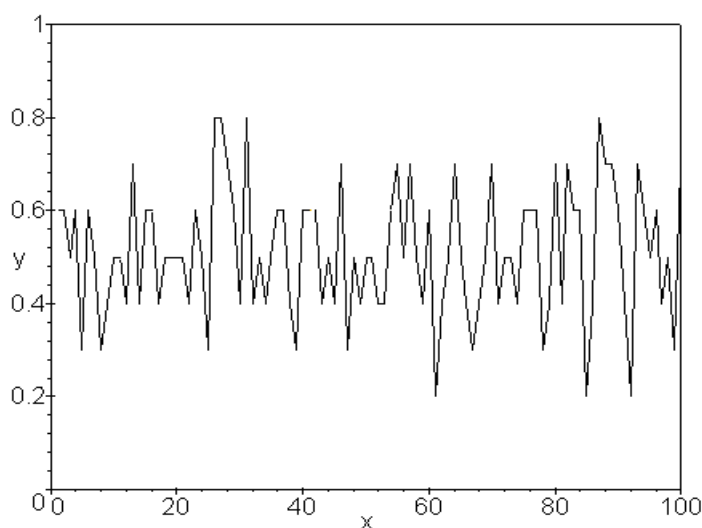
```
> randombit:=rand(0..1);
> for h from 1 to w do s2[h]:=rand();
> for l to t do msg[l]:=rand(): r[l]:=msg[l] mod n;
> x[l]:=r[l]&^2 mod n; b[l]:=randombit(): y[l]:=r[l]*s2[h]^b[l] mod n od:
> seq(x[i],i=1..t): seq(b[i],i=1..t): seq(y[i],i=1..t):
> for l to t do z[l]:=y[l]&^2 mod n; z1[l]:= x[l]*v^b[l] mod n; od:
> seq(z[i],i=1..t): seq(z1[i],i=1..t):
```

Вычисление оценки PR вероятности совпадения в раунде элементов доказательных последовательностей Z и Z1

```
> for i to t do m2[i]:=z[i]-z1[i]; m[i]:=evalf(((z[i]-z1[i]) mod n)/n); c[i] :=1-ceil(m[i]) od:
> C:=seq(c[i],i=1..t):           # Инцидентная последовательность C #
> pr[h]:=sum(c[k], k=1..t)/t od:
> PR:=seq(pr[h],h=1..w);        # Последовательность значений оценки PR #
```

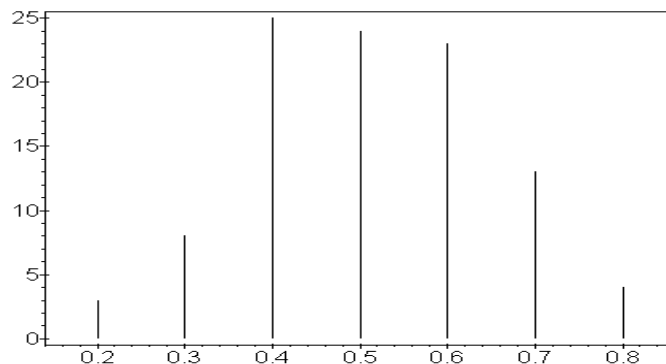
Графическое отображение оценки PR

```
> with(plots): plot({[[k,pr[k]] $k=1..w]}, x=0..w,y=0..1,style=line,numpoints=100);
```



Построение гистограммы для PR


```
> with(stats): with(statplots): histogram([PR]);
```



Программа «Tst 2 FS»

Системные параметры

```
> with(numtheory): with(linalg):
> p:=nextprime(rand()*rand()*rand());
      p := 257047879011769627697933934460816667
> q:=nextprime(rand()*rand()*rand());
      q := 492216721466636908631225730019625269
> n:=p*q;
                                     # Вычисление модуля n #
                                     n :=
126523264267125994185133329451407727584901044025148597231263807049558423
> s:=(rand()*rand()*rand()) mod n;
                                     # Вычисление секретного ключа #
      s := 73107724712145343447046552679000
> igcd(s,n);
                                     1
> v:=s&^2 mod n;
                                     # Вычисление открытого ключа #
      v := 53447394125868269437989819948231690324182898377922077041000000
```

Задание параметров теста: число раундов **t**; число **w** попыток подстановки ложного секретного ключа **s2**; число **d** попыток подбора модуля **n**

```
> w:=10; t:=30; d:=10;
```

Моделирование Tst 2

```
> randombit:=rand(0..1):
> for g to w do
> q1[g]:=rand(); p1[g]:=rand(); n1[g] :=q1[g]*p1[g];
```

```

> for h to d do s2[h]:=rand();
> for l to t do msg[l]:=rand(): r[l]:=msg[l] mod n1[g];
> x[l]:=r[l]&^2 mod n1[g]; b[l]:=randombit(): y[l]:=r[l]*s2[h]^b[l] mod n1[g] od:
> seq(x[i],i=1..t): seq(b[i],i=1..t): seq(y[i],i=1..t):
> for l to t do z[l]:=y[l]&^2 mod n1[g]; z1[l]:= x[l]*v^b[l] mod n1[g]; od:
> seq(z[i],i=1..t): seq(z1[i],i=1..t);

```

Вычисление оценки PR1 вероятности совпадения в раунде элементов доказательных последовательностей Z и Z1

```

> for i to t do m2[i]:=z[i]-z1[i]; m[i]:=evalf(((z[i]-z1[i]) mod n1[g])/n); c[i] :=1-ceil(m[i])
  od; seq(c[i],i=1..t); # Инцидентная последовательность C #
> pr[h]:=sum(c[a], a=1..t)/(t) ; od;
> c1[g]:=seq(pr[k],k=1..d); od:
> PR1:=array(1..w*d): > PR1:=convert(cat[c1[a] $a=1..w],list) :
> PR2:=seq(PR1[i],i=1..w*d);

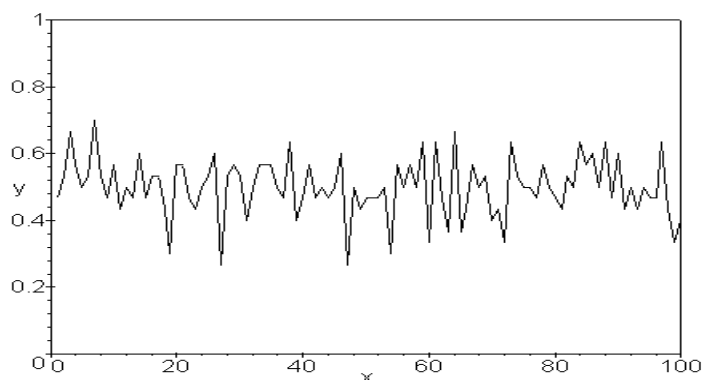
```

Графическое отображение оценки PR1

```

> with(plots):plot({[[a,PR2[a]] $a=1..w*d]},x=0..w*d,y=0..1,style=line,numpoints=100);

```

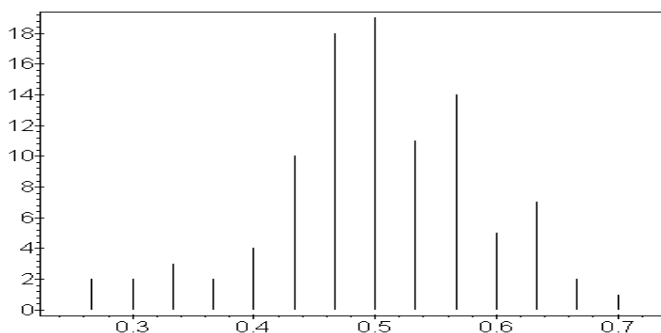


Построение гистограммы PR1

```

> with(stats): with(statplots): histogram([PR2]);

```



Программа «Криптопротокол Гиллоу-Куискуотера»

Системные параметры

```
> with(numtheory): with(linalg):
> p:=nextprime(rand()*rand()); # Вычисление случайного простого числа p #
      p := 13558693922547573765631
> q:=nextprime(rand()*rand()); # Вычисление случайного простого числа q #
      q := 171675368879301488681177
> n:=p*q; # Вычисление модуля n #
      n := 2327693780674897974390356490196290213879227687
> f:=(p-1)*(q-1); # Вычисление функции Эйлера #
      f := 2327693780674897974390171256133488364816780880
> v1:=0: while igcd(v1,f)<>1 do v1:=rand() od:
> if igcd(v1,f)=1 then v:=v1 fi; # Вычисление открытого ключа v #
      v : 432613486829
> igcd(v,f); # Проверка условия взаимной простоты чисел v и f #
      1
> s:=v&^(-1) mod f; # Вычисление секретного ключа s #
      s := 1325273564898245661753935175617017240295093909
```

Выбор параметров пользователя

```
➤ h1:=0: while igcd(h1,f)<>1 do h1:=rand() od:
➤ if igcd(h1,f)=1 and h1<>v then h:=h1; else `return` fi; # Задание идентификацион-
ного номера (хэш-функции) претендента h #
      h := 509974546787
> sa:=h&^(-s) mod n; # Секретный аккредитационный номер претендента sa #
      sa := 357475660958333209920165128744418582701237689
```

Моделирование протокола информационного обмена

```
> r1:=rand():
> if r1<(n-1) then r:=r1; fi; # Вычисление секретного числа пользователем A #
      r := 830196465066
> x:=r&^v mod n; # Вычисление числа x претендентом A #
      x := 1001221517916060202352373217934722793092929974
> e1:=rand(): while e1>v do e1:=rand(); od:
> if e1<v then e:=e1 fi; # Вычисление случайного числа e верификатором B #
```

$e := 151776562728$

> $y := r * sa \&^e \bmod n$; # Вычисление числа y претендентом A #
 $y := 335000620497233576938771090545310137234505100$

Проверка верификатором представленных доказательств

> $z := (h \&^e) * (y \&^v) \bmod n$;
 $z := 1001221517916060202352373217934722793092929974$
> if $z=x$ then 'YES' else 'NO' fi;
YES

Контроль работоспособности программы при действии ложного секретного ключа $s2$

> $s2 := \text{rand}()$;
 $s2 := 166993783344$
> $sa2 := h \&^{(-s2)} \bmod n$;
 $712312218508352562559110564294280957253654467$
> $y := r * sa2 \&^e \bmod n$;
 $y := 1928529588033135094205474699836033259083683850$
> $z := (h \&^e) * (y \&^v) \bmod n$;
 $z := 1483848403250319532429895927813879720267535789$
> if $z=x$ then 'YES' else 'NO' fi;
NO

Комментарии к программам.

1. Подключение библиотеки теории чисел осуществляется с помощью команды **with(numtheory)**. Команды **with(linalg)** и **with(stats)** подключают библиотеки соответственно линейной алгебры и статистики.

2. Программы FS, FFS, GQ моделируют криптопротоколы, соответственно, Фиата-Шамира, Фейге – Фиата – Шамира и Гиллоу-Куискуотера. Модель процесса проверки знаний претендента основана на вычислении и сравнении элементов так называемых доказательных последовательностей $Z = \{z[i], i=1, \dots, t\}$ и $Z1 = \{z1[i], i=1, \dots, t\}$. Элементы $z[i]$ и $z1[i]$ представляют собой числовой результат проверочных вычислений в каждом из t раундов. Аутентификация считается успешной, если элементы доказательных последовательностей совпадают $Z = Z1$. В этом случае программы формируют слово “YES”. В противном случае генерируется слово “NO”.

3. Программа “Tst1 FS” вычисляет статистические характеристики и моделирует атаку на протокол Фиата-Шамира методом подбора секретного ключа при известном значении модуля n . Для этого в блоке программы «Моделирование процесса информационного обмена» формируется случайным образом ложный ключ s_2 , который подставляется вместо секретного ключа s . Количество попыток подстановки ложных значений ключа задается числом w . В каждом раунде программа формирует *инцидентную* последовательность $C = \{c[i]\}$ совпадений или несовпадений элементов доказательных последовательностей.

При этом

$$c[i] = \begin{cases} 1, & \text{если } z[i] = z_1[i] \\ 0, & \text{если } z[i] \neq z_1[i] \end{cases}.$$

В качестве оценки вероятности совпадения элементов доказательных последовательностей в t раундах обмена используются частоты

$$pr[h] = \frac{\sum_{i=1}^t c[i]}{t},$$

которые группируются в статистический ряд PR. Графическое отображение ряда выполняется с помощью команд библиотеки `with(plots)`. Гистограмма ряда строится с помощью команд библиотеки `with(stats)`, `statplots`.

4. Программа “Tst2 FS” вычисляет статистические характеристики и моделирует атаку на протокол Фиата-Шамира методом подбора секретного ключа и модуля n . Количество попыток подстановки ложных значений ключа и модуля задаются числами соответственно w и d . Методика вычисления статистических характеристик такая же как и в программе “Tst1 FS”. Однако статистический ряд PR1 содержит уже $(w \cdot d)$ частотей совпадений элементов доказательных последовательностей в t раундах информационного обмена.

6.СОДЕРЖАНИЕ ОТЧЕТА

1. Формулировка цели работы
2. Схемы исследуемых протоколов аутентификации.
3. Результаты моделирования.
4. Выбор и обоснование наилучших параметров исследуемого протокола.
5. Оценка уровня защиты информации в исследуемых протоколах.
6. Выводы

7. Контрольные вопросы

1. Поясните сущность метода доказательств без разглашения знаний.
2. В чем различие процессов идентификации и аутентификации?
3. Проведите сравнительный анализ исследуемых протоколов.
4. Известно $p = 569$, $q = 34579$. $v = 54955$, $r = 65446$, $e = 38980$. Рассчитать остальные параметры GQ протокола.
5. Какими вероятностными характеристиками оценивается уровень защищенности криптопротокола?
6. Поясните модель криптопротокола на основе машины Тьюринга.
7. Как влияет выбор параметров k и t на вычислительную сложность, длительность сеанса связи и уровень защищенности FFS протокола?
8. Дайте определение квадратичного вычета числа по модулю p и символа Якоби. Какая между ними связь

ЛИТЕРАТУРА

1. Баричев С.Г., Гончаров В.В., Серов Р.Е. Основы современной криптологии. - М.: Горячая линия-Телеком, 2001.-120 с.
2. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях.-М.: КУДИЦ-ОБРАЗ, 2001.-368 с.
3. Введение в криптографию / Под ред. В.В. Яшенко. – 2-е изд. испр. – М.: МЦНМШ «ЧеРо», 1999
4. Menezes F.J., VanOorschot P., Vanstone S. Handbook of Applied Cryptography. – CRC Press, N.Y., 1996.- 780 p.