Федеральное агентство по образованию Российской Федерации
Омский государственный университет им. Ф.М. Достоевского
Факультет компьютерных наук
Кафедра информационной безопасности

Н.Ф. Богаченко

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA

Омск - 2008

УДК 003.26
ББК 32.973-018.1я73
Б 733

Богаченко Н.Ф.

Б 733 Криптографические протоколы на языке программирования Java: Учебно-методическое пособие. - Омск: ОмГУ, 2008. - 36 с.

В пособии представлены практические задания по курсу «Криптографические протоколы». Задания предназначены для выполнения на языке программирования Java, позволяющем существенно упростить реализацию многих алгоритмов шифрования за счет применения класса BigInteger из пакета java.math.

Подробное описание большинства указанных в пособии криптографических протоколов можно найти в работах [1-3, 5, 6].

Предназначается для студентов, обучающихся по специальности 075200 - «Компьютерная безопасность».

УДК 003.26
ББК 32.973-018.1я73

Одобрено учебно-методической комиссией и ученым советом факультета компьютерных наук ОмГУ.

Содержание

Терминология

Протокол – это порядок действий, предпринимаемых двумя или более сторонами, предназначенный для решения определенных задач.

Криптографический протокол – это протокол, использующий криптографию для решения, по крайней мере, одной из задач: обеспечение конфиденциальности, обеспечение целостности, аутентификация и т.д. Иными словами, криптография в протоколе необходима для предотвращения или обнаружения вредительства и мошенничества.

Согласно общепринятой терминологии, в приведенных в пособии криптопротоколах используются следующие обозначения *действующих лиц протокола* [5]:

A (Алиса) - первый участник протокола.

B (Боб) - второй участник протокола.

T (Тренд) - заслуживающий доверия посредник.

Требования к выполнению заданий

1. Задания, если не оговорено противное, выполняются на языке программирования **Java** с использованием класса **BigInteger** из пакета **java.math** (см. п.6), а также пакета **java.net** (см. п.7).

2. Каждый из участников протокола должен быть реализован как отдельный процесс.

3. В протоколах распределения ключей и аутентификации после обмена ключами требуется реализовать шифрование сообщений простым последовательным побитовым наложением сеансового ключа на открытый текст и операцией **xor**. Сообщения представляют собой текст, состоящий из цифр, букв (англ. и рус.) и знаков препинания.

4. Ошибки, которые могут возникнуть в процессе выполнения протокола (несовпадение случайных чисел, имен, порядковых номеров, «просроченные» метки времени и т.п.), должны быть учтены и обработаны (желательно, как исключительные ситуации [4, с.647-661]).

5. Программный продукт должен сопровождаться пояснительной запиской, содержащей подробное описание криптографического протокола и используемых в нем алгоритмов шифрования.

6. Используемый в протоколе симметричный алгоритм шифрования (на общем секретном или сеансовом ключе) оговаривается с преподавателем. В простейшем случае – это гаммирование.

7. Для шифрования с открытым ключом могут быть задействованы следующие криптоалгоритмы:
- алгоритм RSA [5, гл.19.3];
- алгоритм Эль-Гамаля [5, гл.19.6];
- алгоритм Рабина [5, гл.19.5].

8. Для электронной подписи применяется схема RSA.

9. В качестве однонаправленных хэш-функций, зависящих от ключа (кодов проверки подлинности), могут быть использованы следующие алгоритмы:
- метод Джунемана [5, гл.18.14];
- алгоритм проверки подлинности сообщений (Message Authenticator Algorithm, MAA) [5, гл. 18.14];
- алгоритм RSA [5, гл.18.12].

1. Протоколы распределения ключей

Необходимые теоретические сведения

Криптопротокол 1.1.

Алиса и Боб генерируют пары «открытый ключ / закрытый ключ» и открытые ключи отправляют Тренту.

1. Алиса обращается к Тренту и запрашивает сеансовый ключ для связи с Бобом.
2. Трент генерирует случайный сеансовый ключ K и зашифровывает две его копии открытыми ключами Алисы и Боба. Обе копии Трент отправляет Алисе.
3. Алиса расшифровывает свою копию сеансового ключа.
4. Алиса отсылает Бобу его копию сеансового ключа.
5. Боб расшифровывает свою копию сеансового ключа.
6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Криптопротокол 1.2.

Боб генерирует пару «открытый ключ / закрытый ключ» и открытый ключ отправляет Тренту.

1. Алиса получает от Трента открытый ключ Боба.
2. Алиса генерирует случайный сеансовый ключ K, шифрует его открытым ключом Боба и отправляет Бобу.
3. Боб расшифровывает сообщение Алисы своим закрытым ключом.
4. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Криптопротокол 1.3. ***Взаимоблокировка.***

1. Алиса отправляет Бобу свой открытый ключ.
2. Боб отправляет Алисе свой открытый ключ.
3. Алиса генерирует случайное число R_A, шифрует его с помощью открытого ключа Боба. Половину зашифрованного сообщения она отправляет Бобу.
4. Боб генерирует случайное число R_B, шифрует его с помощью открытого ключа Алисы. Половину зашифрованного сообщения он отправляет Алисе.
5. Алиса отправляет Бобу оставшуюся половину зашифрованного сообщения.

6. Боб складывает две половины сообщения Алисы и расшифровывает сообщение Алисы своим закрытым ключом. Затем Боб отсылает Алисе вторую половину своего шифрованного сообщения.

7. Алиса складывает две половины сообщения Боба и расшифровывает сообщение Боба своим закрытым ключом.

8. Алиса и Боб вычисляют сеансовый ключ: $K = R_A \oplus R_B$.

9. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K .

Криптопротокол 1.4.

Боб генерирует пару «открытый ключ / закрытый ключ» и открытый ключ отправляет Тренту.

1. Алиса получает от Трента открытый ключ Боба: K_B^{open} .

2. Алиса генерирует случайный сеансовый ключ K и шифрует сообщение M , используя K : $E_K(M)$.

3. Алиса зашифровывает K , используя открытый ключ Боба: $E_{K_B^{open}}(K)$.

4. Алиса отправляет Бобу как зашифрованное сообщение, так и зашифрованный ключ: $E_K(M), E_{K_B^{open}}(K)$.

5. Боб расшифровывает сеансовый ключ Алисы K своим закрытым ключом.

6. Боб расшифровывает сообщение Алисы с помощью сеансового ключа.

7. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K .

Криптопротокол 1.5. *Encrypted Key Exchange* (*EKE*).

Протокол включает в себя взаимную идентификацию участников. Алиса и Боб имеют общий ключ P .

1. Алиса случайным образом генерирует пару «открытый ключ / закрытый ключ». Далее Алиса шифрует открытый ключ K_A^{open} с помощью симметричного алгоритма, используя P в качестве ключа, и посылает Бобу сообщение: $A, E_P(K_A^{open})$ (A – это имя Алисы).

2. Боб знает P . Он расшифровывает сообщение, получая K_A^{open} . Затем он генерирует случайный сеансовый ключ K , зашифровывает его открытым ключом, который он получил от Алисы, а затем

еще и ключом P. Он посылает Алисе такое сообщение: $E_P(E_{K_A^{open}}(K))$.

3. Алиса расшифровывает сообщение, получает K. Она генерирует случайное число R_A, шифрует его с помощью K и посылает Бобу: $E_K(R_A)$.

4. Боб расшифровывает сообщение, получая R_A. Он генерирует другое случайное число R_B, шифрует оба числа ключом K и посылает Алисе результат: $E_K(R_A, R_B)$.

5. Алиса расшифровывает сообщение, получая R_A и R_B. Если строка R_A, полученная от Боба, - это та самая строка, которую она послала Бобу, Алиса, используя K, шифрует строку R_B и посылает ее Бобу: $E_K(R_B)$.

6. Боб расшифровывает сообщение, получая R_B. Если строка R_B, полученная от Алисы, - это та самая строка, которую он посылал ей, то происходит передача сообщений.

7. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Криптопротокол 1.6. *Diffie-Hellman*.

1. Алиса и Боб вместе выбирают большое простое число n и число g так, чтобы g было примитивным корнем по модулю n.

2. Алиса выбирает случайное большое целое число x и посылает Бобу $X = g^x \bmod n$.

3. Боб выбирает случайное большое целое число y и посылает Алисе $Y = g^y \bmod n$.

4. Алиса вычисляет значение $K = Y^x \bmod n$.

5. Боб вычисляет значение $K' = X^y \bmod n$.

6. Алиса и Боб обмениваются секретными сообщениями с ключом $K = K'$.

Криптопротокол 1.7. *Hughes*.

1. Алиса и Боб вместе выбирают большое простое число n и число g так, чтобы g было примитивным корнем по модулю n.

2. Алиса выбирает случайное большое целое число x и генерирует $K = g^x \bmod n$.

3. Боб выбирает случайное большое целое число y, взаимно простое с $(n-1)$, и посылает Алисе $Y = g^y \bmod n$.

4. Алиса посылает Бобу $X = Y^x \bmod n$.

5. Боб вычисляет значения $z = y^{-1} \bmod (n-1)$ и $K' = X^z \bmod n$.

6. Алиса и Боб обмениваются секретными сообщениями с ключом $K = K'$.

Практические задания

1.1. Напишите консольное приложение, реализующее криптопротокол 1.1. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.2. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.1. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.3. Напишите консольное приложение, реализующее криптопротокол 1.1. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.4. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.1. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.5. Напишите консольное приложение, реализующее криптопротокол 1.1. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.6. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.1. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.7. Напишите консольное приложение, реализующее криптопротокол 1.2. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.8. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.2. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.9. Напишите консольное приложение, реализующее криптопротокол 1.2. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.10. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.2. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.11. Напишите консольное приложение, реализующее криптопротокол 1.2. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.12. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.2. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.13. Напишите консольное приложение, реализующее криптопротокол 1.3. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.14. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.3. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.15. Напишите консольное приложение, реализующее криптопротокол 1.3. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.16. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.3. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.17. Напишите консольное приложение, реализующее криптопротокол 1.3. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.18. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.3. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.19. Напишите консольное приложение, реализующее криптопротокол 1.4. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.20. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.4. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.21. Напишите консольное приложение, реализующее криптопротокол 1.4. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.22. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.4. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.23. Напишите консольное приложение, реализующее криптопротокол 1.4. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.24. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.4. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.25. Напишите консольное приложение, реализующее криптопротокол 1.5. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.26. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.5. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

1.27. Напишите консольное приложение, реализующее криптопротокол 1.5. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.28. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.5. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

1.29. Напишите консольное приложение, реализующее криптопротокол 1.5. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.30. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.5. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

1.31. Напишите консольное приложение, реализующее криптопротокол 1.6.

1.32. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.6.

1.33. Напишите консольное приложение, реализующее криптопротокол 1.7.

1.34. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 1.7.

2. Протоколы аутентификации, использующие хэш-функции и симметричную криптографию

Необходимые теоретические сведения

Протокол 2.1. ***SKID3.***

Это криптографический протокол идентификации, который предполагает, что Алиса и Боб используют общий секретный ключ K.

1. Алиса выбирает случайное число R_A. Она посылает это число Бобу.

2. Боб выбирает случайное число R_B. Он посылает Алисе: $R_B, H_K(R_A, R_B, B)$ (H_K – это хэш-функция, B – имя Боба).

3. Алиса рассчитывает $H_K(R_A, R_B, B)$ и сравнивает результат со значением, полученным от Боба. Если результаты совпадают, Алиса убеждается в том, что она соединилась именно с Бобом.

4. Алиса посылает Бобу: $H_K(R_B, A)$ (A – это имя Алисы).

5. Боб рассчитывает $H_K(R_B, A)$ и сравнивает результат со значением, полученным от Алисы. Если результаты совпадают, Боб убеждается в том, что он соединился именно с Алисой.

6. Алиса и Боб шифруют свои сообщения, используя секретный ключ K.

Протокол 2.2. ***Wide-Mouth Frog.***

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса объединяет метку времени, имя Боба и случайный сеансовый ключ, затем шифрует созданное сообщение общим с Трентом ключом и результат посылает Тренту вместе со своим именем: $A, E_{K_A}(T_A, B, K)$.

2. Трент расшифровывает сообщение от Алисы. Затем он берет новую метку времени, имя Алисы и случайный сеансовый ключ, шифрует полученное сообщение общим с Бобом ключом и посылает результат Бобу: $E_{K_B}(T_T, A, K)$.

3. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Протокол 2.3. ***Yahalom.***

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса объединяет свое имя и случайное число и отправляет созданное сообщение Бобу: A, R_A.

2. Боб объединяет имя Алисы, ее случайное число, свое случайное число, шифрует созданное сообщение общим с Трентом ключом и результат посылает Тренту, добавляя свое имя: $B, E_{K_B}(A, R_A, R_B)$.

3. Трент создает два сообщения. Первое включает имя Боба, случайный сеансовый ключ K, случайные числа Боба и Алисы, оно шифруется ключом, общим для Трента и Алисы. Второе состоит из имени Алисы, случайного сеансового ключа, оно шифруется ключом, общим для Трента и Боба. Трент посылает оба сообщения Алисе: $E_{K_A}(B, K, R_A, R_B)$, $E_{K_B}(A, K)$.

4. Алиса расшифровывает первое сообщение, извлекает K и убеждается, что R_A совпадает со значением, отправленным на этапе 1. Алиса посылает Бобу два сообщения. Одним является сообщение Трента, зашифрованное ключом Боба. Второе - это R_B, зашифрованное сеансовым ключом: $E_{K_B}(A, K)$, $E_K(R_B)$.

5. Боб расшифровывает первое сообщение, извлекает K и убеждается, что R_B совпадает с отправленным на этапе 2.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Протокол 2.4. ***Needham-Schroeder.***

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса посылает Тренту сообщение, содержащее ее имя, имя Боба и случайное число: A, B, R_A.

2. Трент генерирует случайный сеансовый ключ. Он шифрует сообщение, содержащее случайный сеансовый ключ и имя Алисы, секретным ключом, общим для него и Боба. Затем он шифрует случайное число Алисы, имя Боба, ключ, и шифрованное сообщение секретным ключом, общим для него и Алисы. Наконец, он отправляет шифрованное сообщение Алисе: $E_{K_A}(R_A, B, K, E_{K_B}(K, A))$.

3. Алиса расшифровывает сообщение и извлекает K. Она убеждается, что R_A совпадает со значением, отправленным Тренту на этапе 1. Затем она посылает Бобу сообщение, зашифрованное Трентом ключом Боба: $E_{K_B}(K, A)$.

4. Боб расшифровывает сообщение и извлекает K. Затем он генерирует другое случайное число R_B. Он шифрует это число ключом K и отправляет Алисе: $E_K(R_B)$.

5. Алиса расшифровывает сообщение с помощью ключа K. Она создает число $R_B - 1$ и шифрует это число ключом K. Затем она посылает это сообщение обратно Бобу: $E_K(R_B - 1)$.

6. Боб расшифровывает сообщение с помощью ключа K и проверяет значение $R_B - 1$.

7. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Протокол 2.5. ***Otway-Rees.***

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса создает сообщение, состоящее из порядкового номера, ее имени, имени Боба и случайного числа. Сообщение шифруется ключом, общим для Алисы и Трента. Она посылает это сообщение Бобу вместе с порядковым номером, ее и его именами: $I, A, B, E_{K_A}(R_A, I, A, B)$.

2. Боб создает сообщение, состоящее из нового случайного числа, порядкового номера, имени Алисы и имени Боба. Сообщение шифруется ключом, общим для Боба и Трента. Он посылает это сообщение Тренту вместе шифрованным сообщением Алисы, порядковым номером, ее и его именами: $I, A, B, E_{K_A}(R_A, I, A, B), E_{K_B}(R_B, I, A, B)$.

3. Трент генерирует случайный сеансовый ключ. Затем он создает два сообщения. Одно, состоящее из случайного числа Алисы и сеансового ключа, шифруется ключом, общим для него и Алисы. Другое, состоящее из случайного числа Боба и сеансового ключа, шифруется ключом, общим для него и Боба. Он отправляет два этих сообщения вместе с порядковым номером Бобу: $I, E_{K_A}(R_A, K), E_{K_B}(R_B, K)$.

4. Боб отправляет Алисе сообщение, шифрованное ее ключом, и порядковый номер: $I, E_{K_A}(R_A, K)$.

5. Алиса расшифровывает сообщение, получая свои ключ и случайное число. Алиса убеждается, что при выполнении протокола они не изменились.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Протокол 2.6. ***Neuman-Stubblebine.***

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса объединяет свое имя и случайное число, и отправляет созданное сообщение Бобу: A, R_A.

2. Боб объединяет имя Алисы, ее случайное число и метку времени, шифрует созданное сообщение общим с Трентом ключом и результат посылает Тренту, добавляя свое имя и новое случайное число: $B, R_B, E_{K_B}(A, R_A, T_B)$.

3. Трент генерирует случайный сеансовый ключ. Затем он создает два сообщения. Первое включает имя Боба, случайное число Алисы, случайный сеансовый ключ, метку времени и шифруется ключом, общим для Трента и Алисы. Второе состоит из имени Алисы, сеансового ключа, метки времени и шифруется ключом, общим для Трента и Боба. Трент посылает оба сообщения Алисе вместе со случайным числом Боба: $E_{K_A}(B, R_A, K, T_B), E_{K_B}(A, K, T_B), R_B$.

4. Алиса расшифровывает сообщение, зашифрованное ее ключом, извлекает K и убеждается, что R_A совпадает со значением, отправленным на этапе 1. Алиса посылает Бобу два сообщения. Одним является сообщение Трента, зашифрованное ключом Боба. Второе - это R_B, зашифрованное сеансовым ключом: $E_{K_B}(A, K, T_B), E_K(R_B)$.

5. Боб расшифровывает сообщение, зашифрованное его ключом, извлекает K и убеждается, что значения T_B и R_B те же, что и отправленные на этапе 2.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Протокол 2.7. ***Упрощенный Kerberos.***

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса посылает Тренту сообщение со своим именем и именем Боба: A, B.

2. Трент создает сообщение, состоящее из метки времени, времени жизни, случайного сеансового ключа и имени Алисы. Он шифрует сообщение ключом, общим для него и Боба. Затем он объединяет метку времени, время жизни, сеансовый ключ, имя Боба, и шифрует полученное сообщение ключом, общим для него и Алисы. Оба шифрованных сообщения он отправляет Алисе: $E_{K_B}(T_T, L, K, A), E_{K_A}(T_T, L, K, B)$.

3. Алиса проверяет, что сообщение от Трента – текущее. Создает сообщение, состоящее из ее имени и метки времени, шифрует его ключом K и отправляет Бобу. Алиса также посылает Бобу сообщение от Трента, шифрованное ключом Боба: $E_K(A, T_T), E_{K_B}(T_T, L, K, A)$.

4. Боб проверяет, что сообщение от Алисы – текущее. Создает сообщение, состоящее из метки времени плюс единица, шифрует его ключом K и отправляет Алисе: $E_K(T_T + 1)$.

5. Алиса проверяет метку времени.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Практические задания

2.1. Напишите консольное приложение, реализующее криптопротокол 2.1. Для вычисления однонаправленной хэш-функции, зависящей от ключа, используйте алгоритм MAA.

2.2. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.1. Для вычисления однонаправленной хэш-функции, зависящей от ключа, используйте алгоритм MAA.

2.3. Напишите консольное приложение, реализующее криптопротокол 2.1. Для вычисления однонаправленной хэш-функции, зависящей от ключа, используйте метод Джунемана.

2.4. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.1. Для вычисления однонаправленной хэш-функции, зависящей от ключа, используйте метод Джунемана.

2.5. Напишите консольное приложение, реализующее криптопротокол 2.1. Для вычисления однонаправленной хэш-функции, зависящей от ключа, используйте схему RSA.

2.6. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.1. Для вычисления однонаправленной хэш-функции, зависящей от ключа, используйте схему RSA.

2.7. Напишите консольное приложение, реализующее криптопротокол 2.2.

2.8. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.2.

2.9. Напишите консольное приложение, реализующее криптопротокол 2.3.

2.10. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.3.

2.11. Напишите консольное приложение, реализующее криптопротокол 2.4.

2.12. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.4.

2.13. Напишите консольное приложение, реализующее криптопротокол 2.5.

2.14. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.5.

2.15. Напишите консольное приложение, реализующее криптопротокол 2.6.

2.16. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.6.

2.17. Напишите консольное приложение, реализующее криптопротокол 2.7.

2.18. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 2.7.

3. Протоколы аутентификации, использующие криптографию с открытым ключом

Необходимые теоретические сведения

Криптопротокол 3.1. *DASS.*

Алиса и Боб генерируют пары «открытый ключ / закрытый ключ» и открытые ключи отправляют Тренту. Трент отправляет Алисе и Бобу свой открытый ключ.

1. Алиса посылает Тренту сообщение, состоящее из имени Боба: B.

2. Трент посылает Алисе открытый ключ Боба K_B^{open}, подписанный закрытым ключом Трента K_T^{close}. Подписанное сообщение содержит имя Боба: $S_{K_T^{close}}(B, K_B^{open})$.

3. Алиса проверяет подпись Трента, убеждаясь, что она действительно получила открытый ключ Боба. Она генерирует случайный сеансовый ключ K и случайную пару «открытый ключ / закрытый ключ» K_P^{open} / K_P^{close}. Она шифрует метку времени T_A ключом K, а затем подписывает время жизни L, свое имя A и сгенерированный открытый ключ K_P^{open} своим закрытым ключом K_A^{close}. Наконец, она зашифровывает K открытым ключом Боба K_B^{open} и подписывает его с помощью K_P^{close}. Все это она отправляет Бобу: $E_K(T_A), S_{K_A^{close}}(L, A, K_P^{open}), S_{K_P^{close}}(E_{K_B^{open}}(K))$.

4. Боб посылает Тренту сообщение, состоящее из имени Алисы: A.

5. Трент посылает Бобу открытый ключ Алисы K_A^{open}, подписанный закрытым ключом Трента K_T^{close}. Подписанное сообщение содержит имя Алисы: $S_{K_T^{close}}(A, K_A^{open})$.

6. Боб проверяет подпись Трента, убеждаясь, что он действительно получил открытый ключ Алисы. Затем он проверяет подпись Алисы и извлекает K_P^{open}. Боб использует свой закрытый ключ и извлекает K. Затем он расшифровывает T_A, проверяя, что это сообщение - текущее.

7. Если требуется обоюдная проверка подлинности, Боб шифрует новую метку времени T_B ключом K и результат посылает Алисе: $E_K(T_B)$.

8. Алиса расшифровывает T_B ключом K, проверяя, что это сообщение - текущее.

9. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Криптопротокол 3.2. ***Denning-Sacco.***

Алиса и Боб генерируют пары «открытый ключ / закрытый ключ» и открытые ключи отправляют Тренту. Трент отправляет Алисе и Бобу свой открытый ключ.

1. Алиса посылает Тренту сообщение, состоящее из ее имени и имени Боба: A, B.

2. Трент посылает Алисе открытый ключ Боба K_B^{open}, подписанный закрытым ключом Трента K_T^{close}. Трент также посылает Алисе ее собственный открытый ключ K_A^{open}, подписанный закрытым ключом Трента K_T^{close}: $S_{K_T^{close}}(B, K_B^{open}), S_{K_T^{close}}(A, K_A^{open})$.

3. Алиса посылает Бобу случайный сеансовый ключ K и метку времени T_A, подписав их своим закрытым ключом K_A^{close} и зашифровав открытым ключом Боба K_B^{open}, вместе с обоими подписанными ключами: $E_{K_B^{open}}(S_{K_A^{close}}(K, T_A)), S_{K_T^{close}}(B, K_B^{open}), S_{K_T^{close}}(A, K_A^{open})$.

4. Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа K_B^{close} и проверяет подпись Алисы с помощью ее открытого ключа K_A^{open}. Он также убеждается, что метка времени правильна.

5. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Криптопротокол 3.3. ***Woo-Lam.***

Алиса и Боб генерируют пары «открытый ключ / закрытый ключ» и открытые ключи отправляют Тренту. Трент отправляет Алисе и Бобу свой открытый ключ.

1. Алиса посылает Тренту сообщение, состоящее из ее имени и имени Боба: A, B.

2. Трент посылает Алисе открытый ключ Боба K_B^{open}, подписанный закрытым ключом Трента K_T^{close}: $S_{K_T^{close}}(B, K_B^{open})$.

3. Алиса проверяет подпись Трента. Затем она посылает Бобу свое имя A и случайное число R_A, шифрованное открытым ключом Боба K_B^{open}: $A, E_{K_B^{open}}(R_A)$.

4. Боб посылает Тренту свое имя B, имя Алисы A и случайное число Алисы R_A, шифрованное открытым ключом Трента K_T^{open}: $A, B, E_{K_T^{open}}(R_A)$.

5. Трент посылает Бобу открытый ключ Алисы K_A^{open}, подписанный закрытым ключом Трента K_T^{close}. Он также посылает Бобу случайное число Алисы R_A, случайный сеансовый ключ K, имена Алисы и Боба A, B, подписав все это закрытым ключом Трента K_T^{close} и зашифровав открытым ключом Боба K_B^{open}: $S_{K_T^{close}}(K_A^{open}), E_{K_B^{open}}(S_{K_T^{close}}(R_A, K, A, B))$.

6. Боб проверяет подписи Трента. Затем он посылает Алисе вторую часть сообщения Трента, полученного на этапе 5, и новое случайное число R_B, зашифровав все открытым ключом Алисы K_A^{open}: $E_{K_A^{open}}(S_{K_T^{close}}(R_A, K, A, B), R_B)$.

7. Алиса проверяет подпись Трента и свое случайное число. Затем она посылает Бобу его случайное число R_B, шифрованное сеансовым ключом K: $E_K(R_B)$.

8. Боб расшифровывает свое случайное число и проверяет, что оно не изменилось.

9. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Практические задания

3.1. Напишите консольное приложение, реализующее криптопротокол 3.1. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

3.2. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.1. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

3.3. Напишите консольное приложение, реализующее криптопротокол 3.1. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

3.4. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.1. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

3.5. Напишите консольное приложение, реализующее криптопротокол 3.1. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

3.6. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.1. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

3.7. Напишите консольное приложение, реализующее криптопротокол 3.2. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

3.8. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.2. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

3.9. Напишите консольное приложение, реализующее криптопротокол 3.2. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

3.10. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.2. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

3.11. Напишите консольное приложение, реализующее криптопротокол 3.2. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

3.12. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.2. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

3.13. Напишите консольное приложение, реализующее криптопротокол 3.3. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

3.14. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.3. В качестве алгоритма шифрования с открытым ключом используйте схему RSA.

3.15. Напишите консольное приложение, реализующее криптопротокол 3.3. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

3.16. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.3. В качестве алгоритма шифрования с открытым ключом используйте схему Эль-Гамаля.

3.17. Напишите консольное приложение, реализующее криптопротокол 3.3. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

3.18. Напишите приложение с графическим интерфейсом, реализующее криптопротокол 3.3. В качестве алгоритма шифрования с открытым ключом используйте схему Рабина.

Дополнительные требования к выполнению заданий

Задания **1.1** – **1.30** выполняются на языке программирования **Java** или **C++**. При этом следующие операции необходимо реализовать и протестировать самостоятельно:

- возведение целого числа в степень в кольце вычетов [5, гл.11.3];
- вычисление наибольшего общего делителя двух целых чисел (алгоритм Евклида) [5, гл.11.3];
- вычисление обратного значения в кольце вычетов (расширенный алгоритм Евклида) [5, гл.11.3];
- генерация большого простого числа (теста Рабина - Миллера) [5, гл.11.5].

Практические задания

5.1. Реализуйте криптографический протокол разделения секрета, использующий схему Блэкли (схема 3 из n).

Комментарий. Схема Блэкли:

Предварительные шаги:

1) выбрать простое p;

2) секрет $x_0 \in Z_p$;

3) случайно выбрать $y_0, z_0 \in Z_p$;

4) получить секретную точку $M = (x_0, y_0, z_0)$.

Разделение секрета:

1) для каждого участника случайно выбрать $a, b \in Z_p$;

2) вычислить $c = z_0 - ax_0 - by_0 (\mathrm{mod}\ p)$;

3) получить плоскость $z = ax + by + c \ (\mathrm{mod}\ p)$;

4) вручить каждому участнику долю секрета (a, b, c).

Восстановление секрета:

Найти точку пересечения трех плоскостей.

5.2. Реализуйте криптографический протокол разделения секрета, использующий схему Шамира (схема t из n). Параметр $t = \mathrm{const}$.

Комментарий. Схема Шамира:

Предварительные шаги:

1) выбрать простое p;

2) секрет $m \in Z_p$;

3) случайно выбрать $s_1, ..., s_{t-1} \in Z_p$;

4) установить $s(x) = m + s_1 x + ... + s_{t-1} x^{t-1} (\mathrm{mod}\ p)$.

Разделение секрета:

1) для каждого участника случайно выбрать $x_i \in Z_p$;

2) вычислить $s(x_i)$;

3) вручить каждому участнику долю секрета $(x_i, s(x_i))$.

Восстановление секрета:

$$m = \sum_{i=1}^{t} s(x_i) \frac{\prod_{i \neq j} x_j}{\prod_{i \neq j} (x_j - x_i)} \pmod{p}.$$

5.3. Реализуйте криптографический протокол из задания 5.2 для произвольного t.

5.4. Реализуйте криптографический протокол для конференц-связи группы из трех участников, использующий модифицированную схему Диффи-Хелмана.

Комментарий. Описание криптопротокола можно найти в работе [1, с.402-403].

5.5. Реализуйте криптографический протокол для конференц-связи группы из t участников.

Комментарий. Описание криптопротокола можно найти в работе [1, с.403-404].

5.6. Реализуйте криптографический протокол «подбрасывания монеты по телефону», использующий схему «привязки к биту».

Комментарий. Описание криптопротокола «привязки к биту»:

1. Алиса и Боб вместе выбирают большое простое число n и число g так, чтобы g было примитивным корнем по модулю n.

2. Боб выбирает случайное число q и посылает Алисе $y = g^q \pmod{n}$.

3. Алиса выбирает случайное число k, затем подбрасывает монету и результат b в «связанном состоянии» посылает Бобу: $y^b g^k \pmod{n}$.

4. Алиса посылает Бобу k и b. Боб проверяет, что на шаге 3 Алиса вручила действительно b.

Чтобы модифицировать эту схему в протокол «подбрасывания монеты по телефону», необходимо добавить следующие пункты:

3,5. Боб посылает Алисе свою догадку b'.

6. Алиса и Боб вычисляют $b \oplus b'$.

5.7. Реализуйте протокол о «византийских генералах» (протокол BG) для упрощенной задачи «командир и его заместители».

Комментарий. Криптопротокол строится индукцией по числу предателей. Пусть n - число участников протокола (1 командир и n-1 его заместителей); m - максимальное число предателей.

Протокол BG(0)

1. Командир рассылает заместителям приказ.
2. Заместители поступают в соответствии с полученным приказом.

Протокол BG(m)

1. Командир рассылает заместителям приказ.
2. Каждый заместитель рассылает этот приказ своим коллегам, используя протокол BG(m -1). На время рассылки рассылающий выступает в роли командира для своих коллег.
3. Каждый заместитель из (n -1)-го приказа (1 приказ свой и (n -2) приказа, полученных от коллег) выбирает наиболее часто встречающийся и поступает в соответствии с ним.

5.8. Реализуйте криптографический протокол «игры в покер по телефону» для двух игроков на основе коммутирующих криптосистем.

Комментарий. Коммутирующие криптосистемы можно построить по схеме RSA над общим Z_n, так как $(e^{t_1})^{t_2} = (e^{t_2})^{t_1} \pmod{n}$. Описание криптопротокола:

1. Алиса и Боб создают пары «открытый ключ /закрытый ключ»: Алиса - K_A^{open} / K_A^{close}, Боб - K_B^{open} / K_B^{close}.
2. Алиса зашифровывает своим открытым ключом все 52 карты в колоде, перемешивает и результат отправляет Бобу.
3. Боб случайным образом выбирает из них 5 шифрограмм для Алисы и отправляет их ей.
4. Алиса с помощью своего закрытого ключа расшифровывает свои карты.
5. Боб случайным образом из оставшихся шифрограмм выбирает 5 для себя, зашифровывает их своим открытым ключом и отправляет Алисе.
6. Алиса расшифровывает их своим закрытым ключом и отправляет Бобу.
7. Боб окончательно расшифровывает свои карты своим закрытым ключом.
8. Если требуется добирать карты из колоды, то поступают аналогичным образом.
9. В конце игры Алиса и Боб раскрывают свои карты и пары ключей для того, чтобы каждый мог убедиться в отсутствии мошенничества.

5.9. Реализуйте криптографический протокол из задания 5.8 для трех игроков.

Комментарий. Описание криптопротокола можно найти в работе [5, гл.4.11].

5.10. Реализуйте криптографический протокол доказательства с нулевым разглашением на основе задачи «Изоморфизм графов».

Комментарий. Описание криптопротокола можно найти в работе [6, с.37].

5.11. Реализуйте криптографический протокол доказательства с нулевым разглашением на основе задачи «Гамильтонов цикл».

Комментарий. Описание криптопротокола можно найти в работе [5, гл.5.1].

5.12. Реализуйте криптографический протокол доказательства с нулевым разглашением на основе задачи «3-раскраска графа».

Комментарий. Описание криптопротокола «3-раскраска графа»:

1. Доказывающий случайным образом переставляет 3 цвета между собой, шифрует цвета всех вершин (например, используя привязку к биту) и отсылает проверяющему.
2. Проверяющий выбирает случайную пару вершин, соединенных ребром.
3. Доказывающий открывает цвета этих вершин.
4. Проверяющий убеждается, что цвета разные.

Шаги 1-4 повторяются n раз.

5.13. Реализуйте криптографический протокол Фиата-Шамира доказательства с нулевым разглашением.

Комментарий. Описание криптопротокола можно найти в работе [1, с.341-343].

5.14. Реализуйте криптографический протокол слепой подписи с выборочной проверкой.

Комментарий. Описание криптопротокола можно найти в работе [5, гл.5.3].

В качестве функции подписи и маскирующего множителя возможно использовать схему RSA: пусть e - открытый ключ, d - закрытый ключ, M - подписываемый документ, R - случайное число. Тогда $M^d = M^d RR^{-1} = (MR^e)^d R^{-1} (\mathrm{mod}\, n)$, где $R^e (\mathrm{mod}\, n)$ - маскирующий множитель.

6. Примеры реализации алгоритмов шифрования на Java

В данном разделе приведены примеры использования класса **BigInteger** из пакета **java.math** языка **Java** для реализации алгоритмов шифрования. Класс **BigInteger** предназначен для работы с целыми числами произвольной длины [4].

Для реализации симметричного алгоритма гаммирования можно воспользоваться встроенным методом **xor**. Этот метод выполняет побитовое сложение по модулю 2 (не заботясь о том, что ключ может быть короче текста).

```
int size = 1024;
Random r = new Random();

// Генерация случайного целого неотрицательного числа,
// равномерно распределенного в диапазоне от 0 до 2^size – 1
BigInteger Key = new BigInteger(size,r);

// Открытый текст
String S = "Криптографические протоколы";

// Перевод двоичного числа, содержащегося
// в байтовом массиве S.getBytes(), в большое целое число
BigInteger BI = new BigInteger(S.getBytes());

// Побитовый XOR
BI = BI.xor(Key);

// Шифротекст в виде большого целого числа
System.out.println(BI.toString());
// Шифротекст в виде строки
System.out.println(new String(BI.toByteArray()));
```

Необходимо учитывать, что если первый символ строки **S** принадлежит кириллице, то сопоставляемое ей число в формате **BigInteger**, при использовании конструктора **BigInteger**(**S.getBytes**()), будет отрицательным. В приведенном примере этот факт не играет роли. Но при реализации других криптоалгоритмов, например RSA, отрицательность числа, сопоставляемого открытому тексту, приведет к ошибке. Решить эту проблему можно, например, приписывая в начало строки «фиктивный» символ.

Методы класса **BigInteger** существенно облегчают реализацию алгоритмов шифрования с открытым ключом. В качестве примера приведем фрагмент программного кода, демонстрирующий генерацию пары «открытый / закрытый ключ» криптоалгоритма RSA.

```
int size = 2048;
Random r  = new Random();

// Генерация случайных целых неотрицательных чисел p и q,
// которые в двоичном представлении занимают size бит
// и с вероятностью 1-1/2^10000 являются простыми
BigInteger p = new BigInteger(size,10000,r);
BigInteger q = new BigInteger(size,10000,r);

// Вычисление n = pq
BigInteger n = BigInteger.ONE;
n = n.multiply(p).multiply(q);

// Вспомогательные расчеты
BigInteger p1 = BigInteger.ONE;
p1 = p1.negate().add(p);
BigInteger q1 = BigInteger.ONE;
q1 = q1.negate().add(q);
BigInteger n1 = BigInteger.ONE;
n1 = n1.multiply(p1).multiply(q1);

// Выбор случайного числа e (открытый ключ),
// взаимнопростого с (p-1)(q-1)
BigInteger e = new BigInteger(size,r);
while(e.gcd(n1).compareTo(BigInteger.ONE) != 0)
        e = new BigInteger(size,r);

// Вычисление числа d (закрытый ключ),
// обратного к числу e по модулю (p-1)(q-1)
BigInteger d = BigInteger.ONE;
d = e.modInverse(n1);
```

Процесс зашифрования (или расшифрования) теперь заключается в вызове метода **modPow**:

```
BI = BI.modPow(e,n);
```

7. Элементы сетевого программирования на Java

В данном разделе приведены примеры клиент-серверных приложений на языке **Java** с использованием пакета **java.net**.

Рассмотрим реализацию консольного приложения. Сервер запускает два потока: на прием и на отправку сообщений. То же самое следует сделать на стороне клиента. В результате сервер и клиент смогут обмениваться информацией, вводимой с клавиатуры.

В примере представлены варианты обмена строковыми сообщениями и сообщениями в виде потока байт. Последнее может быть полезным при реализации шифрованного обмена.

Строка *exit* интерпретируется как команда о завершении работы. Входящие сообщения и информация об ошибках выводятся на консоль.

Для упрощения программного кода мы пренебрегаем «правилами хорошего тона» и не производим «финальное рукопожатие» при завершении сеанса связи одним из участников протокола.

```
// Подключение библиотек Java
import java.io.*;
import java.net.*;

public class Main {

   Socket s_soket;
   int port =13;
   DataOutputStream s_out;
   DataInputStream s_input;
   boolean b;

   public static void main(String[] args) {
      new Main().startServer();
   }

   public void startServer(){
      try {
         // Создание серверного сокета
         ServerSocket ss = new ServerSocket(port);
         // Прием соединения от клиента
         s_soket = ss.accept();
         // На стороне клиента: s_soket = new Socket(iaddr,port);
```

```
// Создание исходящего потока для сокета
s_out = new DataOutputStream(s_soket.getOutputStream());

// Создание входящего потока для сокета
s_input = new DataInputStream(s_soket.getInputStream());

// Отправка сообщения
s_out.writeUTF("Проверка связи"+"\n");

// Получение сообщения
System.out.println(">"+s_input.readUTF());

b = true;

// Обработка входящего байтового потока
new Thread() {
@Override
public void run() {
   int bsize;
   byte buf[] = new byte[1000];

   try{

      while(b){
         bsize = s_input.read(buf);
         if (bsize = = -1){
            b = false;
            System.out.println("Соединение закрыто другим
                                    участником протокола.");
         } else{
            System.out.println(">"+ new String(buf,0,bsize));
            System.out.println();
         }
      }

      s_soket.close();
      System.exit(0);

   } catch (Exception e){
      // Обработка ошибок чтения
      if (b){
```

```
            System.out.println("Ошибка чтения.");
            System.exit(-1);
        }
    }
}
}.start();

// Обработка исходящего байтового потока
new Thread() {
@Override
public void run() {
    String mes;

    try{
        // Создание буферизированного символьного потока для
        // ввода с консоли
        BufferedReader s_fromConsol = new
                BufferedReader(new InputStreamReader(System.in));

        while(b){
            System.out.println();
            mes = s_fromConsol.readLine();
            if ((mes.compareToIgnoreCase("exit")!=0)){
                s_out.write(mes.getBytes());
            } else{
                b = false;
                System.out.println("Соединение закрыто.");
            }
        }

        s_soket.close();
        System.exit(0);

    } catch (Exception e){
        // Обработка ошибок записи
        System.out.println("Ошибка записи.");
        System.exit(-1);
    }
}
}.start();
```

```
        } catch (Exception e){
            // Обработка ошибок открытия сокета
            System.out.println("Ошибка установки соединения.");
            System.exit(-1);
        }
    }
}
```

В случае реализации приложения с графическим интерфейсом, класс **Thread** следует использовать только для обработки входящего потока.

Отметим, что с целью упрощения программного кода, в приведенном ниже примере не обработаны ситуации корректного завершения сеанса связи и закрытия соединения другим участником протокола, а также не детализированы ошибки в блоках **try / catch**.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// Кнопка "прием соединения от клиента"
        try {
            ServerSocket ss = new ServerSocket(port);
            s_socet = ss.accept();
            s_out = new DataOutputStream(s_socet.getOutputStream());
            s_out.writeUTF("Проверка связи." + "\n");
            s_input = new DataInputStream(s_socet.getInputStream());
            new Thread() {
                @Override
                public void run() {
                    try {
                        while (true) {
                            jTextArea1.append(">"+s_input.readUTF());
                        }
                    } catch (Exception e) {
                        System.err.println(e);
                    }
                }
            }.start();
        } catch (IOException e){
            System.err.println(e);
        }
}
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// Кнопка "отправить сообщение"
        try {
            s_out.writeUTF(jTextField1.getText()+"\n");
        } catch (Exception e){
            System.err.println(e);
        }
}
```

Литература

1. Алферов А.П., Зубков А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. - М.: Гелиос АРВ, 2001. - 480 с.

2. Белим С.В. Задания по курсу «Криптографические протоколы». - Омск: ОмГУ, 2005. - 26 с.

3. Саломаа А. Криптография с открытым ключом. - М.: Мир, 1995. - 320 с.

4. Хорстманн К.С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы. - М.: Издательский дом «Вильямс», 2003. - 848 с.

5. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. - М .: Триумф, 2002. - 816 с.

6. Ященко В.В. Введение в криптографию. - СПб.: Питер, 2001. - 288 с.

Надежда Федоровна Богаченко

Криптографические протоколы
на языке программирования Java

Авторское редактирование

Подписано в печать 30.09.2008.
ОП. Формат 60x84 1/16 . Усл.печ.л. 2,25. Уч.-изд.л. 2,15.
Тираж 100 экз.