

БДЗ по прикладной криптографии

Фирсов Георгий, М21-507

9 мая 2022 г.

Содержание

Задание 1	2
Задание 2	2
Задание 3	3
Задание 4	3
Задание 5	3
Задание 6	4
Задание 7	5
Задание 8	5
Задание 9	5
Задание 10	6
Задание 11	7
Задание 12	8

Задание 1

Анна генерирует два числа $x \xleftarrow{R} \mathbb{Z}_1, y \xleftarrow{R} \mathbb{Z}_q$, после чего отправляет Борису тройку $(A_0, A_1, A_2) = (g^x, g^y, g^{xy+a})$.

Борис генерирует свои два числа $r \xleftarrow{R} \mathbb{Z}_q, s \xleftarrow{R} \mathbb{Z}_q$, а затем отправляет Анне следующую пару: $(B_1, B_2) = (A_1^r \cdot g^s, (A_2/g^b)^r \cdot A_0^s)$. Заметим, что:

$$\begin{aligned} B_1 &= A_1^r \cdot g^s = g^y \cdot g^s = g^{y+s} \\ B_2 &= (A_2/g^b)^r \cdot A_0^s = g^{xy+a} \cdot g^{-b} \cdot g^{xs} = g^{x(y+s)+a-b} \end{aligned}$$

Если B_1 возвести в степень x и затем умножить на обратный к полученному элемент число B_2 , то получится g^{a-b} :

$$\begin{aligned} B_1^x &= (g^{y+s})^x = g^{x(y+s)} \\ B_2 \cdot (B_1^x)^{-1} &= g^{x(y+s)+a-b} \cdot g^{-x(y+s)} = g^{a-b} \end{aligned}$$

Если $a = b$, то $g^{a-b} = g^0 = e_{\mathbb{G}}$. Это свойство и можно использовать для проверки равенства чисел a и b .

Ответ: в) Анна проверяет равенство $B_2/B_1^x = 1$.

Задание 2

Так как числа p, a, b общеизвестны, то считаю, что при разработке программы все *возможные* вычисления с данными параметрами выполняются заранее (то есть, собственно, на этапе разработки программы). Несложно увидеть, что:

$$\begin{aligned} H^{(n)}(x) &= \underbrace{H_{p,a,b}(H_{p,a,b}(\cdots H_{p,a,b}(x) \cdots))}_{n \text{ раз}} = \underbrace{a(a(\cdots ax + b \cdots) + b)}_{n \text{ раз}} + b = \\ &= a^n x + b \sum_{j=0}^{n-1} a^j \pmod p \end{aligned}$$

Обозначим:

$$\begin{aligned} a' &:= a^n \pmod p \\ b' &:= b \sum_{j=0}^{n-1} a^j \pmod p \end{aligned}$$

Тогда:

$$H^{(n)}(x) = a'x + b' \pmod p$$

Значения a' и b' возможно вычислить предварительно на этапе разработки программы, что позволит вычислять функцию $H^{(n)}$ так же быстро, как и $H_{p,a,b}$.

Но может случиться так, что числа p, a, b *заранее* не известны (например, меняются с течением времени). Таким образом, возникает потребность поддержки вычисления a', b' на лету. В таком случае заметим, что:

$$\sum_{j=0}^{n-1} a^j = (a^n - 1) \cdot (a - 1)^{-1} \pmod p$$

При больших n возведение в степень $n - 1$ потребует примерно столько же операций, сколько и возведение в степень n . Заметим, что возведение в степень можно производить, пользуясь следующей идеей: $a^4 = a^2 \cdot a^2, a^8 = a^4 \cdot a^4$ и т.д. Данный алгоритм требует асимптотически $\log_2(n)$ умножений.

Ответ: а) $H^{(n)}$ может быть вычислена так же быстро, как $H_{p,a,b}$ (в случае известных заранее значений p, a, b); г) вычисление $H^{(n)}$ требует времени $O(\log n)$ (в случае неизвестных заранее значений p, a, b).

Задание 3

Рассмотрим по очереди все варианты, отобрав подходящие:

- $p_1 = (k_1, k_2), p_2 = (k'_1), p_3 = (k'_2)$: владельцы долей p_2, p_3 не смогут вдвоем восстановить ключ, так как $k'_1 \oplus k'_2 = ???$.
- $p_1 = (k_1, k_2), p_2 = (k_2, k'_2), p_3 = (k'_2)$: владелец p_2 может один восстановить ключ, так как $k = k_2 \oplus k'_2$.
- $p_1 = (k_1, k_2), p_2 = (k_1, k_2), p_3 = (k'_2)$: владельцы p_1, p_2 не смогут восстановить вдвоем ключ, так как никакая комбинация k_1, k_2 в сумме не даст k .
- $p_1 = (k_1, k_2), p_2 = (k'_1, k'_2), p_3 = (k'_2)$: владельцы p_2, p_3 не смогут восстановить вдвоем ключ, так как никакая комбинация k'_1, k'_2 в сумме не даст k .
- $p_1 = (k_1, k_2), p_2 = (k'_2), p_3 = (k'_1, k_2)$: данный вариант подходит, так как:
 - p_1, p_2 : $k_2 \oplus k'_2 = k$
 - p_1, p_3 : $k_1 \oplus k'_1 = k$
 - p_2, p_3 : $k'_2 \oplus k_2 = k$

При этом восстановление ключа ни одним участником единолично невозможно, так как ни один из них не обладает двумя частями с одинаковыми индексами.

Ответ: д) $p_1 = (k_1, k_2), p_2 = (k'_2), p_3 = (k'_1, k_2)$

Задание 4

Задание 5

1. Так как каждому участнику $B_i, i \in \{1, \dots, n\}$ известен ключ k , то, например, B_2 может создать некоторое сообщение и рассчитать имитовставку для него с использованием этого ключа. В таком случае участник B_1 , получив сообщение, не может удостовериться, что оно создано участником A , а не B_2 .
2. При условии, что участники $B_i, i \in \{1, \dots, n\}$ не вступают друг с другом в сговор, атака из п. 1 становится неприменимой, если:

$$\forall i, j \in \{1, \dots, n\} : i \neq j \implies S_i \setminus S_j \neq \emptyset \quad (1)$$

Данное условие позволяет каждому участнику B_i иметь хотя бы один такой ключ k_m , которого нет у некоторого другого участника $B_j, j \neq i$. Такой ключ найдется у каждого B_i для каждого B_j . Таким образом, вероятность того, что подделанная имитовставка для данного ключа окажется верной, является пренебрежимо малой при условии, что метод расчета имитовставки является стойким.

В то же время участник A имеет все ключи и может рассчитать все имитовставки корректно.

Таблица 1: Состав подмножеств $S_j, j \in \{1, \dots, 10\}$

Подмножество	k_1	k_2	k_3	k_4	k_5
S_1	+	+	+	—	—
S_2	+	+	—	+	—
S_3	+	+	—	—	+
S_4	+	—	+	+	—
S_5	+	—	—	+	+
S_6	+	—	+	—	+
S_7	—	+	+	+	—
S_8	—	+	+	—	+
S_9	—	+	—	+	+
S_{10}	—	—	+	+	+

- В таблице 1 с помощью знака + обозначено вхождение ключа $k_j, j \in \{1, \dots, 5\}$ в подмножество $S_i, i \in \{1, \dots, 10\}$, а знак — обозначает отсутствие ключа в подмножестве. Несложно заметить, что любые попарные разности являются непустыми, так как все подмножества имеют одинаковую мощность, равную трем, и попарно различны. Таким образом, условие (1) выполнено, что обозначает неприменимость атаки из п. 1 к данной системе (см. п. 2).
- Пусть в сговор вступают, например, B_1 и B_9 . Заметим, что эти два участника вместе имеют в наличии все ключи k_1, \dots, k_5 . Если они обмениваются недостающими ключами, то каждый из них может в таком случае рассчитать имитовставку для произвольного сообщения на каждом ключе и тем самым какой-либо другой участник не может быть уверен в том, что сообщение пришло от A , а не от B_1 или B_9 .

Задание 6

Анне и Борису известны следующие величины:

- v
- $u = g^\alpha v^{-i}$
- Набор $u_j = uv^j = g^\alpha v^{j-i}, j \in \{1, \dots, n\}$

Только Борису известны α и индекс i . Заметим, что для $j = i$: $u_i = g^\alpha$.

Анна пересылает Борису следующие значения:

$$(a_j, b_j) = (g^{k_j}, m_j u_j^{k_j}), k_j \xleftarrow{R} \{1, 2, \dots, q-1\}$$

- Восстановление сообщения m_i из полученных данных.* Борис может расшифровать значение m_i (так как α — закрытый ключ криптосистемы Эль-Гамала, соответствующий открытому ключу u_i):

$$b_i a_i^{-\alpha} = m_i g^{\alpha k_i} g^{-k_i \alpha} = m_i$$

- Невозможность получения индекса i Анной.* Попробовать получить значение i Анна может из значений $u = g^\alpha v^{-i}$ и $u_j = g^\alpha v^{j-i}$. Так как $v \in \mathbb{G}$, то $v = g^k$ для некоторого k . $u = g^{\alpha - ki}, u_j = g^{\alpha + k(j-i)}$.

Заметим, что Анне неизвестны значения α, g^α . Отличить $u_i = g^\alpha$ (даже если бы он был известен) от какого-то иного элемента группы вычислительно сложно, то есть перебирая j невозможно понять, когда u_j сравнивается с g^α (то есть при $j = i$).

Получить же i из значения u также вычислительно трудно, так как это предполагает нахождение дискретного логарифма (по $u = g^{\alpha - ki}$ найти $\alpha - ki$ трудно).

Задание 7

Задание 8

Задание 9

Так как $\text{ord } \mathbb{G} = q$ – простое число, то все элементы группы, не равные $e_{\mathbb{G}}$, являются образующими. Это значит, что взяв случайный неединичный элемент группы, гарантированно можно получить образующий.

1. Общеизвестными параметрами схемы мультикоммитмента являются: простое число q , группа \mathbb{G} порядка q , а также случайные (h, g_1, \dots, g_n) , для которых выполнено:

$$\begin{aligned} h &\in \mathbb{G}, h \neq e_{\mathbb{G}} \\ \forall i \in \{1, \dots, n\} : g_i &\in \mathbb{G}, g_i \neq e_{\mathbb{G}} \\ \forall i, j \in \{1, \dots, n\} : g_i &\neq h; i \neq j \implies g_i \neq g_j \end{aligned}$$

Пусть $r \xleftarrow{R} \mathbb{Z}_q$. Функция Commit_n выглядит следующим образом:

$$\text{Commit}_n(\mathbf{m}, r) = h^r \prod_{j=1}^n g_j^{m_j}, \mathbf{m} = (m_1, \dots, m_n) \quad (2)$$

Несложно показать свойство гомоморфизма:

$$\begin{aligned} \text{Commit}_n(\mathbf{m} + \mathbf{m}', r + r') &= h^{r+r'} \prod_{j=1}^n g_j^{m_j+m'_j} = h^r h^{r'} \left(\prod_{j=1}^n g_j^{m_j} \right) \left(\prod_{j=1}^n g_j^{m'_j} \right) = \\ &= \left(h^r \prod_{j=1}^n g_j^{m_j} \right) \left(h^{r'} \prod_{j=1}^n g_j^{m'_j} \right) = \text{Commit}_n(\mathbf{m}, r) + \text{Commit}_n(\mathbf{m}', r') \end{aligned}$$

2. Покажем, что схема, описанная в п. 1 обеспечивает совершенное сокрытие. Для этого следует показать, что для любых векторов $\mathbf{m} = (m_1, \dots, m_n)$ и $\mathbf{m}' = (m'_1, \dots, m'_n)$ равны распределения следующих случайных величин:

$$\begin{aligned} C &= \text{Commit}_n(\mathbf{m}, r), r \xleftarrow{R} \mathbb{Z}_q \\ C' &= \text{Commit}_n(\mathbf{m}', r'), r' \xleftarrow{R} \mathbb{Z}_q \end{aligned}$$

где Commit_n определена согласно (2).

Обозначим через $\alpha_i, i \in \{1, \dots, n\}$ такие числа, что $g_i^{\alpha_i} = h$ (такое число всегда найдется, так как g_i – образующий элемент группы \mathbb{G}). Положим также:

$$\alpha = \sum_{j=1}^n \alpha_j m_j, \alpha' = \sum_{j=1}^n \alpha_j m'_j$$

Пусть $r \xleftarrow{R} \mathbb{Z}_q$, тогда $c = \text{Commit}_n(\mathbf{m}, r) = h^{r+\alpha}$ и $c' = \text{Commit}_n(\mathbf{m}', r) = h^{r+\alpha'}$ (c – розыгрыш случайной величины C , c' – величины C').

Из вычислительной сложности задачи дискретного логарифмирования следует, что c и c' отличить друг от друга так же вычислительно сложно, из чего, в свою очередь, следует равенство распределений случайных величин C и C' .

3. В п. 1 утверждалось, что $h, g_i, i \in \{1, \dots, n\}$ – случайные неединичные элементы группы \mathbb{G} . Также в примечании перед п. 1 утверждается, что любой неединичный элемент из \mathbb{G} является образующим. Упомянутые элементы группы можно получить следующим образом:

$$\begin{aligned} h &= H(i_1) \\ g_j &= H(i_{j+1}), j \in \{1, \dots, n\} \end{aligned} \quad (3)$$

где $H : \mathbb{Z}_q \rightarrow \mathbb{G}$ – случайный оракул, а последовательность $\{i_j\}_{j=1}^{n+1}$ составляется следующим образом:

$$\begin{aligned} i_1 &= k_0 \\ i_{j+1} &= i_j + k_j \end{aligned}$$

где k_0 – минимальное положительное число, при котором $H(k_0) \neq e_{\mathbb{G}}$, k_j – минимальное положительное число, при котором $H(i_j + k_j) \neq e_{\mathbb{G}}$.

Таким образом, элементы, сгенерированные по формулам в (3) являются случайными образующими элементами группы \mathbb{G} , вероятность совпадений среди которых мала. Т.е. они удовлетворяют требованиям из п. 1.

Свойство сокрытия показано в п. 2. Свойство связывания следует из вычислительной сложности задачи логарифмирования: из этого следует отсутствие эффективного алгоритма для генерации таких $r, r', \mathbf{m}, \mathbf{m}'$, что $\text{Commit}_n(\mathbf{m}, r) = \text{Commit}_n(\mathbf{m}', r')$.

Задание 10

1. *Некорректное хэширование.* Так как требуется подделка подписи для любого сообщения, то рассматривается модель стойкости UUF-CMA (Universal Unforgeability under Chosen Message Attack). Рассмотрим противника \mathcal{A} , действующего по следующему сценарию:

- \mathcal{A} генерирует некоторое сообщение $m \in \mathcal{M}$ (любое).
- \mathcal{A} вычисляет хэш сообщения: $c = H(m)$.
- \mathcal{A} вычисляет значение $R = h^{-c}$ ($h = pk$), значение z полагается равным 0.
- Подпись $\sigma = (R, z) = (pk^{-H(m)}, 0)$ отправляется оракулу проверки подписи.

Заметим, что проверка этой подписи выполнится всегда:

$$\left. \begin{aligned} c &\leftarrow H(m) \\ g^z &= e_{\mathbb{G}} \\ R \cdot h^c &= h^{-H(m)} \cdot h^{H(m)} = h^0 = e_{\mathbb{G}} \end{aligned} \right\} \implies V(pk, m, \sigma) = \text{accept}$$

Таким образом:

$$\text{Adv}_{\text{SS}}^{\text{UUF-CMA}}(\mathcal{A}) = \Pr[V(pk, m, \sigma) = \text{accept}] = 1$$

2. *Некорректная генерация случайных чисел.* Обозначим $c_i = H(m_i, R_i), i \in \{0, 1\}$ и, зная, что $\rho_1 = a\rho_0 + b$, запишем:

$$\begin{cases} z_0 = \rho_0 + c_0\alpha \\ z_1 = a\rho_0 + b + c_1\alpha \end{cases}$$

$$\begin{cases} az_0 = a\rho_0 + ac_0\alpha \\ z_1 = a\rho_0 + b + c_1\alpha \end{cases}$$

Тогда:

$$\begin{aligned} z_1 - az_0 &= b + \alpha(c_1 - ac_0) \\ \alpha &= (z_1 - az_0 - b)(c_1 - ac_0)^{-1} \end{aligned}$$

В общем-то $(c_1 - ac_0)$ – некоторый случайный элемент из \mathbb{Z}_q , а так как q – простое число, то для обратимости элемента требуется только то, чтобы он был ненулевым. Вероятность того, что элемент $(c_1 - ac_0)$ обратим, равна:

$$\Pr \left[\begin{array}{c} \text{Элемент } (c_1 - ac_0) \\ \text{обратим в кольце } \mathbb{Z}_q \end{array} \right] = 1 - \frac{1}{|\mathbb{Z}_q|} = \frac{q-1}{q} \xrightarrow{q \rightarrow \infty} 1$$

3. *Атака по взаимосвязанным ключам.* Пусть $pk_i = h \cdot g^i = g^\alpha \cdot g^i = g^{\alpha+i}$, тогда через обозначим $sk_i = \alpha_i = \alpha + i$ (согласно нотации криптосистемы). Введем также обозначение $\Delta\alpha_{ij} := j - i$.

Пусть для некоторого сообщения m имеется некоторая подпись $\sigma_i = (R_i, z_i)$, где $z_i = \rho_i + c_i\alpha_i$, $c_i = H(m, R_i)$. Противник генерирует следующую подпись: $\sigma_j = (R_i, z_j)$, где $z_j = \rho_i + c_i\alpha_i + c_i\Delta\alpha_{ij} = \rho_i + c_i\alpha_j$.

Эта подпись является валидной для того же сообщения и успешно проверяется на ключе pk_j :

$$\left. \begin{aligned} c &= H(m, R_i) \\ g^{z_j} &= g^{\rho_i + c_i\alpha_j} = g^{\rho_i + c_i(\alpha+j)} \\ R_i \cdot pk_j^c &= g^{\rho_i} \cdot g^{c(\alpha+j)} = g^{\rho_i + c(\alpha+j)} \end{aligned} \right\} \implies V(pk_j, m, \sigma_j) = \text{аccept}$$

Задание 11

1. Пусть пассивный противник записал успешный сеанс протокола аутентификации, то есть в его распоряжении находятся запрос m и ответ t . при этом также известно, что $t = MAC(sk, m)$ для некоторого sk , являющегося в данном случае паролем.

Перебирая sk по словарю и сравнивая получаемые значения имитовставки от m , рассчитанной на перебираемом значении в качестве ключа, с t , противник может найти подходящее значение sk . Ясно, что успешность атаки зависит от размера словаря и самого пароля, однако словарная атака на такую схему аутентификации в принципе возможна.

2. Пусть активный противник как бы “подменяет” сервер и отдает клиенту свой сертификат $Cert_E$. Если клиент не проверяет сертификат должным образом, то он будет считать, что общается с сервером, хотя на самом деле обмен пакетами происходит с противником¹.

Сгенерировав запрос² и получив ответ, противник получает в распоряжение пару m, t запроса и ответа, которая может использоваться для проведения словарной атаки (см. п. 1).

¹Противник при этом может проксировать запросы на настоящий сервер для сокрытия своего присутствия. Это возможно, если он сам установит соединение с сервером.

²Это не понадобится, если противник проксирует запросы/ответы с/на сервер. Кроме того, такое проксирование позволит убедиться в правильности ответа клиента.

3. Проведем некоторые преобразования:

$$U_2 = (S/h^{pw})^u = (g^s h^{pw}/h^{pw})^u = g^{su}$$

$$U_1 U_2 = g^u k^{pw} g^{su} = g^{u(s+1)} k^{pw}$$

Сервер получает $U_1 = g^u k^{pw}$ и $U_3 = H(pw, S, U_1 U_2)$. При этом серверу требуется рассчитать хэш от набора значений $pw, S, U_1 U_2$ и сравнить с полученным U_3 . S и pw серверу известны, надо посчитать значение $U_1 U_2$, что можно сделать следующим образом:

$$U_1 U_2 = g^{u(s+1)} k^{pw} = \frac{g^{u(s+1)} k^{pw} k^{s \cdot pw}}{k^{s \cdot pw}} = \frac{g^{u(s+1)} k^{pw \cdot (s+1)}}{k^{s \cdot pw}} = \frac{(g^u k^{pw})^{s+1}}{k^{s \cdot pw}} = \frac{U_1^{s+1}}{k^{s \cdot pw}}$$

что возможно, так как s и k серверу так же известны.

Таким образом, на сервере осуществляется проверка ниже, а сеанс аутентификации считается успешным, если она пройдена.

$$U_3 \stackrel{?}{=} H\left(pw, S, \frac{U_1^{s+1}}{k^{s \cdot pw}}\right)$$

Задание 12