Лекция 7. Протоколы вручения обязательства (commitment protocols)

Косолапов Ю.В.

ЮФУ

15 октября 2020 г.

Содержание

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : b_A$ (выбранное значение передается в открытом виде);

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : b_A$ (выбранное значение передается в открытом виде);
- B выбирает случайно значение $b_B \in \{0, 1\}$;

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : b_A$ (выбранное значение передается в открытом виде);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : b_A$ (выбранное значение передается в открытом виде);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);
- $A, B : b = b_A \oplus b_B$ (вычисляется «сторона монеты»);

Пример: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : b_A$ (выбранное значение передается в открытом виде);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);
- $A, B : b = b_A \oplus b_B$ (вычисляется «сторона монеты»);

Вопрос

Какие проблемы у этого протокола?

Задача: подбрасывание монеты по телефону

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.

Протокол:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : b_A$ (выбранное значение передается в открытом виде);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);
- $A, B : b = b_A \oplus b_B$ (вычисляется «сторона монеты»);

Ответ

B всегда может выбрать выигрышное значение $b_B = 1 - b_A$.

Определение

Нестрогое определение СВО

Участник протокола A намерен передать/вручить участнику протокола B на хранение (**commit**) некое значение **a**, но не намерен сразу же раскрывать (**reveal**) это значение **a** (то есть **a** передается/вручается в зашифрованном виде **c**). **Фаза вручения** и **фаза раскрытия** распределены во времени.

Определение

Нестрогое определение СВО

Участник протокола A намерен передать/вручить участнику протокола B на хранение (**commit**) некое значение **a**, но не намерен сразу же раскрывать (**reveal**) это значение **a** (то есть **a** передается/вручается в зашифрованном виде **c**). **Фаза вручения** и **фаза раскрытия** распределены во времени.

Требования:

Определение

Нестрогое определение СВО

Участник протокола A намерен передать/вручить участнику протокола B на хранение (**commit**) некое значение **a**, но не намерен сразу же раскрывать (**reveal**) это значение **a** (то есть **a** передается/вручается в зашифрованном виде **c**). **Фаза вручения** и **фаза раскрытия** распределены во времени.

Требования:

- участник B не должен по **c** получить информацию об **a**, пока не наступила фаза раскрытия;

Определение

Нестрогое определение СВО

Участник протокола A намерен передать/вручить участнику протокола B на хранение (**commit**) некое значение **a**, но не намерен сразу же раскрывать (**reveal**) это значение **a** (то есть **a** передается/вручается в зашифрованном виде **c**). **Фаза вручения** и **фаза раскрытия** распределены во времени.

Требования:

- участник B не должен по **c** получить информацию об **a**, пока не наступила фаза раскрытия;
- участник A не должен иметь возможности подменить значение **a** во время фазы раскрытия.

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол на основе СВО:

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол на основе СВО:

- A выбирает случайно значение $b_A \in \{0, 1\}$;

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.

Протокол на основе СВО:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : \mathbf{c} = \mathrm{COMMIT}(b_A)$ (выбранное значение прячется в $\mathbf{c}$ и передается B);

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол на основе СВО:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : \mathbf{c} = \mathrm{COMMIT}(b_A)$ (выбранное значение прячется в $\mathbf{c}$ и передается B);
- B выбирает случайно значение $b_B \in \{0, 1\}$;

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол на основе СВО:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : \mathbf{c} = \mathrm{COMMIT}(b_A)$ (выбранное значение прячется в $\mathbf{c}$ и передается B);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол на основе СВО:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : \mathbf{c} = \mathrm{COMMIT}(b_A)$ (выбранное значение прячется в $\mathbf{c}$ и передается B);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);
- $A \to B : b_A$ (фаза раскрытия: A раскрывает значение, отправленное ранее);

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону[1]

Условие: если выпадает 0 (орел), то побеждает A, если 1 (решка), то побеждает B.
Протокол на основе СВО:

- A выбирает случайно значение $b_A \in \{0, 1\}$;
- $A \to B : \mathbf{c} = \mathrm{COMMIT}(b_A)$ (выбранное значение прячется в $\mathbf{c}$ и передается B);
- B выбирает случайно значение $b_B \in \{0, 1\}$;
- $B \to A : b_B$ (выбранное значение передается в открытом виде);
- $A \to B : b_A$ (фаза раскрытия: A раскрывает значение, отправленное ранее);
- $A, B : b = b_A \oplus b_B$ (вычисляется «сторона монеты» или общий случайный бит);

[1] Другое название: совместная генерация случайного бита

Пример 1: подбрасывание монеты по телефону (Проблемы)

Проблема 1

B генерирует свой ответ (орел или решка) после того, как получит $\mathbf{c} = \mathrm{COMMIT}(b_A)$. Поэтому B может попытаться получить некую информацию по $\mathbf{c}$ о значении b_A.

Пример 1: подбрасывание монеты по телефону (Проблемы)

Проблема 1

B генерирует свой ответ (орел или решка) после того, как получит $\mathbf{c} = \mathrm{COMMIT}(b_A)$. Поэтому B может попытаться получить некую информацию по $\mathbf{c}$ о значении b_A.

Решение: преобразование COMMIT должно быть таким, чтобы по $\mathbf{c} = \mathrm{COMMIT}(b_A)$ участник B не получал информацию о b_A.

Пример 1: подбрасывание монеты по телефону (Проблемы)

Проблема 1

B генерирует свой ответ (орел или решка) после того, как получит $\mathbf{c} = \mathrm{COMMIT}(b_A)$. Поэтому B может попытаться получить некую информацию по $\mathbf{c}$ о значении b_A.

Решение: преобразование COMMIT должно быть таким, чтобы по $\mathbf{c} = \mathrm{COMMIT}(b_A)$ участник B не получал информацию о b_A.

Проблема 2

A, получив b_B, может попытаться отправить $1 - b_A$ (если $b_A \oplus b_B \neq 0$) вместо b_A.

Пример 1: подбрасывание монеты по телефону (Проблемы)

Проблема 1

B генерирует свой ответ (орел или решка) после того, как получит $\mathbf{c} = \mathrm{COMMIT}(b_A)$. Поэтому B может попытаться получить некую информацию по $\mathbf{c}$ о значении b_A.

Решение: преобразование COMMIT должно быть таким, чтобы по $\mathbf{c} = \mathrm{COMMIT}(b_A)$ участник B не получал информацию о b_A.

Проблема 2

A, получив b_B, может попытаться отправить $1 - b_A$ (если $b_A \oplus b_B \neq 0$) вместо b_A.

Решение: преобразование COMMIT должно быть таким, чтобы $\mathbf{c} = \mathrm{COMMIT}(b_A)$ соответствовало только b_A.

Пример 2: организация торгов

Пример 2: организация торгов

- B – организатор конкурса на поставку мела в школу;

Пример 2: организация торгов

- B – организатор конкурса на поставку мела в школу;
- A – участник конкурса (возможный поставщик мела);

Пример 2: организация торгов

- B – организатор конкурса на поставку мела в школу;
- A – участник конкурса (возможный поставщик мела);
- есть и другие поставщики (участники конкурса): C, D;

Пример 2: организация торгов

- B – организатор конкурса на поставку мела в школу;
- A – участник конкурса (возможный поставщик мела);
- есть и другие поставщики (участники конкурса): C, D;
- Все участники подают свои коммерческие предложения (заявки): z_A, z_C, z_D.

Пример 2: организация торгов

- B – организатор конкурса на поставку мела в школу;
- A – участник конкурса (возможный поставщик мела);
- есть и другие поставщики (участники конкурса): C, D;
- Все участники подают свои коммерческие предложения (заявки): z_A, z_C, z_D.
- Обычно на сбор заявок выделяется некоторое время (не все сразу).

Пример 2: организация торгов

- B – организатор конкурса на поставку мела в школу;
- A – участник конкурса (возможный поставщик мела);
- есть и другие поставщики (участники конкурса): C, D;
- Все участники подают свои коммерческие предложения (заявки): z_A, z_C, z_D.
- Обычно на сбор заявок выделяется некоторое время (не все сразу).
- В конкурсе побеждает тот, кто привезет мел по наименьшей цене (понятно, что поставщики должны при этом также получить прибыль).

Пример 2: организация торгов (проблемы)

Проблема 1

Организатор B может оказаться недобросовестным: он может по поступившим заявкам z_A, z_C, z_D определить минимальную цену, и сообщить своему другу E, чтобы он подал заявку z_E с ценой, на рубль меньшей минимальной цены. Тогда победит E.

Пример 2: организация торгов (проблемы)

Проблема 1

Организатор B может оказаться недобросовестным: он может по поступившим заявкам z_A, z_C, z_D определить минимальную цену, и сообщить своему другу E, чтобы он подал заявку z_E с ценой, на рубль меньшей минимальной цены. Тогда победит E.

Решение: заявки подаются в опечатанных конвертах, конверты раскрываются в присутствии всех участников (в назначенное время).

Пример 2: организация торгов (проблемы)

Проблема 1

Организатор B может оказаться недобросовестным: он может по поступившим заявкам z_A, z_C, z_D определить минимальную цену, и сообщить своему другу E, чтобы он подал заявку z_E с ценой, на рубль меньшей минимальной цены. Тогда победит E.

Решение: заявки подаются в опечатанных конвертах, конверты раскрываются в присутствии всех участников (в назначенное время).

Проблема 2

Участник, например, A может договориться с B, чтобы его конверт с заявкой вскрывался последним. Тогда A может подготовить несколько вариантов предложений, а B в момент вскрытия конверта A (когда уже известна минимальная цена по предыдущим конвертам) подменяет заявку z_A заявкой z'_A с подходящей для выигрыша ценой.

Пример 2: организация торгов (проблемы)

Проблема 1

Организатор B может оказаться недобросовестным: он может по поступившим заявкам z_A, z_C, z_D определить минимальную цену, и сообщить своему другу E, чтобы он подал заявку z_E с ценой, на рубль меньшей минимальной цены. Тогда победит E.

Решение: заявки подаются в опечатанных конвертах, конверты раскрываются в присутствии всех участников (в назначенное время).

Проблема 2

Участник, например, A может договориться с B, чтобы его конверт с заявкой вскрывался последним. Тогда A может подготовить несколько вариантов предложений, а B в момент вскрытия конверта A (когда уже известна минимальная цена по предыдущим конвертам) подменяет заявку z_A заявкой z'_A с подходящей для выигрыша ценой.

Решение: проводить видео-запись вскрытия конвертов (следить за B).

Строгое определение

A – отправитель, B – получатель.

Определение

Пусть $\mathrm{COMMIT} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$ — детерминированное отображение (k – параметр безопасности, e.g. длина ключа) полиномиальной сложности (т.е. простое в вычислении). Схема вручения обязательства состоит из двух протоколов:

Строгое определение

A – отправитель, B – получатель.

Определение

Пусть $\text{COMMIT} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$ — детерминированное отображение (k – параметр безопасности, e.g. длина ключа) полиномиальной сложности (т.е. простое в вычислении). Схема вручения обязательства состоит из двух протоколов:

- **Фаза вручения**: отправитель для значения $x \in \{0,1\}^*$ вычисляет обязательство $\mathbf{c} = \text{COMMIT}(u, x)$, где ключ u выбирается случайно и равновероятно из $\{0,1\}^k$. Далее $\mathbf{c}$ передается получателю. Получатель сохраняет у себя $\mathbf{c}$ для дальнейшего использования. Т.о.

$$A \to B : \mathbf{c} = \text{COMMIT}(u, x).$$

Строгое определение

A – отправитель, B – получатель.

Определение

Пусть $\mathrm{COMMIT} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$ — детерминированное отображение (k – параметр безопасности, e.g. длина ключа) полиномиальной сложности (т.е. простое в вычислении). Схема вручения обязательства состоит из двух протоколов:

- **Фаза вручения**: отправитель для значения $x \in \{0,1\}^*$ вычисляет обязательство $\mathbf{c} = \mathrm{COMMIT}(u, x)$, где ключ u выбирается случайно и равновероятно из $\{0,1\}^k$. Далее $\mathbf{c}$ передается получателю. Получатель сохраняет у себя $\mathbf{c}$ для дальнейшего использования. Т.о.

$$A \to B : \mathbf{c} = \mathrm{COMMIT}(u, x).$$

- **Фаза раскрытия**: отправитель для передает u', x' (честный передаст u, x, а нечестный может подменить значения). Получатель вычисляет $\mathrm{COMMIT}(u', x')$ и сравнивает с $\mathbf{c}$:

Строгое определение

A – отправитель, B – получатель.

Определение

Пусть $\text{COMMIT} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$ — детерминированное отображение (k – параметр безопасности, e.g. длина ключа) полиномиальной сложности (т.е. простое в вычислении). Схема вручения обязательства состоит из двух протоколов:

- **Фаза вручения**: отправитель для значения $x \in \{0,1\}^*$ вычисляет обязательство $\mathbf{c} = \text{COMMIT}(u, x)$, где ключ u выбирается случайно и равновероятно из $\{0,1\}^k$. Далее $\mathbf{c}$ передается получателю. Получатель сохраняет у себя $\mathbf{c}$ для дальнейшего использования. Т.о.

$$A \to B : \mathbf{c} = \text{COMMIT}(u, x).$$

- **Фаза раскрытия**: отправитель для передает u', x' (честный передаст u, x, а нечестный может подменить значения). Получатель вычисляет $\text{COMMIT}(u', x')$ и сравнивает с $\mathbf{c}$:

$$(\text{COMMIT}(u', x') \stackrel{?}{=} \mathbf{c}) = \begin{cases} \textit{True}, \text{ A не обманывает} \\ \textit{False}, \text{ A обманывает.} \end{cases}$$

Строгое определение

A – отправитель, B – получатель.

Определение

Пусть $\text{COMMIT} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$ — детерминированное отображение (k – параметр безопасности, e.g. длина ключа) полиномиальной сложности (т.е. простое в вычислении). Схема вручения обязательства состоит из двух протоколов:

- **Фаза вручения**: отправитель для значения $x \in \{0,1\}^*$ вычисляет обязательство $\mathbf{c} = \text{COMMIT}(u, x)$, где ключ u выбирается случайно и равновероятно из $\{0,1\}^k$. Далее $\mathbf{c}$ передается получателю. Получатель сохраняет у себя $\mathbf{c}$ для дальнейшего использования. Т.о.

$$A \to B : \mathbf{c} = \text{COMMIT}(u, x).$$

- **Фаза раскрытия**: отправитель для передает u', x' (честный передаст u, x, а нечестный может подменить значения). Получатель вычисляет $\text{COMMIT}(u', x')$ и сравнивает с $\mathbf{c}$:

$$(\text{COMMIT}(u', x') \stackrel{?}{=} \mathbf{c}) = \begin{cases} \textit{True}, \text{ A не обманывает} \\ \textit{False}, \text{ A обманывает.} \end{cases}$$

Если $x \in \{0,1\}$, то говорят о **схеме вручения бита** (bit ciommitment scheme).

Требования безопасности

Чтобы рассмотреть, какие требования выдвигаются к СВБ, необходимо ввести некоторые дополнительные понятия.

Пренебрежимо малая функция

Неотрицательная функция $f : \mathbb{N} \to \mathbb{R}$ называется **пренебрежимо малой**, если для каждого $\gamma \in \mathbb{N}$ найдется $k_0 \in \mathbb{N}$, такой, что для всех $k \geqslant k_0$: $f(k) \leqslant \frac{1}{k^\gamma}$.

Требования безопасности

Чтобы рассмотреть, какие требования выдвигаются к СВБ, необходимо ввести некоторые дополнительные понятия.

Пренебрежимо малая функция

Неотрицательная функция $f : \mathbb{N} \to \mathbb{R}$ называется **пренебрежимо малой**, если для каждого $\gamma \in \mathbb{N}$ найдется $k_0 \in \mathbb{N}$, такой, что для всех $k \geqslant k_0$: $f(k) \leqslant \frac{1}{k^\gamma}$.

Example

Пренебрежимо малые: $f(k) = 2^{-k}$, $f(k) = 2^{-\sqrt{k}}$.
Не пренебрежимо малые: $f(k) = k^{-1}$, $f(k) = k^{-1000}$.

Требования безопасности

Чтобы рассмотреть, какие требования выдвигаются к СВБ, необходимо ввести некоторые дополнительные понятия.

Пренебрежимо малая функция

Неотрицательная функция $f : \mathbb{N} \to \mathbb{R}$ называется **пренебрежимо малой**, если для каждого $\gamma \in \mathbb{N}$ найдется $k_0 \in \mathbb{N}$, такой, что для всех $k \geqslant k_0$: $f(k) \leqslant \frac{1}{k^\gamma}$.

Example

Пренебрежимо малые: $f(k) = 2^{-k}$, $f(k) = 2^{-\sqrt{k}}$.
Не пренебрежимо малые: $f(k) = k^{-1}$, $f(k) = k^{-1000}$.

Статистическое расстояние между случайными величинами

Пусть X, Y — случайные величины с областью значений V. Статистическим расстоянием называется величина:

$$\Delta(X, Y) = \frac{1}{2} \sum_{v \in V} |\Pr\{X = v\} - \Pr\{Y = v\}|.$$

Требования безопасности к СВБ

Требования безопасности к СВБ

- Требование **связности** (binding) — для любого атакующего E вероятность нахождения таких u и u', что

$$\textsc{commit}(u, 0) = \textsc{commit}(u', 1)$$

должна быть ничтожной/пренебрежимо малой (negligible) функцией по k. Например:

$$\Pr\{u, u' \leftarrow E : \textsc{commit}(u, 0) = \textsc{commit}(u', 1)\} \sim \frac{1}{2^k}$$

Требования безопасности к СВБ

- Требование **связности** (binding) — для любого атакующего E вероятность нахождения таких u и u', что

$$\textsc{commit}(u, 0) = \textsc{commit}(u', 1)$$

должна быть ничтожной/пренебрежимо малой (negligible) функцией по k. Например:

$$\Pr\{u, u' \leftarrow E : \textsc{commit}(u, 0) = \textsc{commit}(u', 1)\} \sim \frac{1}{2^k}$$

- Требование **маскировки** (hiding) — распределения случайных величин $\textsc{commit}(U, 0)$ и $\textsc{commit}(U, 1)$ должны быть **неразличимы** атакующим E, когда случайная величина U принимает значения равновероятно из $\{0, 1\}^k$.

Пример СВБ

Пусть $x \in \{0,1\}$, $k = 128$, $u \in_R \mathbb{Z}_{2^k}$,

$$\text{COMMIT}(u, x) = u + x(\text{MOD}2^k),$$

где суммирование выполняется в кольце $\mathbb{Z}_{2^k}$

Пример СВБ

Пусть $x \in \{0,1\}$, $k = 128$, $u \in_R \mathbb{Z}_{2^k}$,

$$\text{COMMIT}(u, x) = u + x(\text{MOD}2^k),$$

где суммирование выполняется в кольце $\mathbb{Z}_{2^k}$
Проверим свойства связности и маскировки:

Пример СВБ

Пусть $x \in \{0,1\}$, $k = 128$, $u \in_R \mathbb{Z}_{2^k}$,

$$\text{COMMIT}(u, x) = u + x(\text{MOD}2^k),$$

где суммирование выполняется в кольце $\mathbb{Z}_{2^k}$
Проверим свойства связности и маскировки:

- Требование связности не выполняется, так как, например, $\text{COMMIT}(135, 0) = \text{COMMIT}(134, 1)$.

Пример СВБ

Пусть $x \in \{0,1\}$, $k = 128$, $u \in_R \mathbb{Z}_{2^k}$,

$$\text{COMMIT}(u, x) = u + x(\text{MOD}2^k),$$

где суммирование выполняется в кольце $\mathbb{Z}_{2^k}$
Проверим свойства связности и маскировки:

- Требование связности не выполняется, так как, например, $\text{COMMIT}(135, 0) = \text{COMMIT}(134, 1)$. Такие «коллизии» найти легко.

Пример СВБ

Пусть $x \in \{0,1\}$, $k = 128$, $u \in_R \mathbb{Z}_{2^k}$,

$$\text{COMMIT}(u, x) = u + x(\text{MOD}2^k),$$

где суммирование выполняется в кольце $\mathbb{Z}_{2^k}$
Проверим свойства связности и маскировки:

- Требование связности не выполняется, так как, например, $\text{COMMIT}(135, 0) = \text{COMMIT}(134, 1)$. Такие «коллизии» найти легко.
- Требование маскировки выполняется, так как случайные величины $\text{COMMIT}(U, 0)$ и $\text{COMMIT}(U, 1)$ распределены равномерно над $\mathbb{Z}_{2^k}$.

Вычислительная и теоретико-информационная связность

Определение вычислительной связности

Требование **связности** (binding) — для любого **полиномиально ограниченного** атакующего E вероятность нахождения таких u и u', что

$$\textsc{commit}(u, 0) = \textsc{commit}(u', 1)$$

должна быть ничтожной/пренебрежимо малой (negligible) функцией по k.

Вычислительная и теоретико-информационная связность

Определение вычислительной связности

Требование **связности** (binding) — для любого **полиномиально ограниченного** атакующего E вероятность нахождения таких u и u', что

$$\textsc{commit}(u, 0) = \textsc{commit}(u', 1)$$

должна быть ничтожной/пренебрежимо малой (negligible) функцией по k.

Атакующий может потратить на подбор коллизии (u, u') порядка $p(k)$ единиц времени и порядка $q(k)$ единиц памяти (полиномы $p(k)$ и $q(k)$ для каждого атакующего, в общем случае, свои).

Вычислительная и теоретико-информационная связность

Определение вычислительной связности

Требование **связности** (binding) — для любого **полиномиально ограниченного** атакующего E вероятность нахождения таких u и u', что

$$\textsc{commit}(u, 0) = \textsc{commit}(u', 1)$$

должна быть ничтожной/пренебрежимо малой (negligible) функцией по k.

Атакующий может потратить на подбор коллизии (u, u') порядка $p(k)$ единиц времени и порядка $q(k)$ единиц памяти (полиномы $p(k)$ и $q(k)$ для каждого атакующего, в общем случае, свои).

Определение теоретико-информационной связности

Если на атакующего E не накладываются ограничения на вычислительную способность и объем памяти, то говорят о **теоретико-информационной связности**.

Вычислительная и теоретико-информационная маскировка

Определение вычислительной маскировки

Требование **вычислительной маскировки** (hiding) — распределения случайных величин $\mathrm{COMMIT}(U, 0)$ и $\mathrm{COMMIT}(U, 1)$ должны быть неразличимы **полиномиально ограниченным** атакующим E, когда случайная величина U принимает значения равновероятно из $\{0,1\}^k$.

Вычислительная и теоретико-информационная маскировка

Определение вычислительной маскировки

Требование **вычислительной маскировки** (hiding) — распределения случайных величин $\text{COMMIT}(U, 0)$ и $\text{COMMIT}(U, 1)$ должны быть неразличимы **полиномиально ограниченным** атакующим E, когда случайная величина U принимает значения равновероятно из $\{0, 1\}^k$.

Атакующий может потратить на различение распределений порядка $p'(k)$ единиц времени и порядка $q'(k)$ единиц памяти (полиномы $p'(k)$ и $q'(k)$ для каждого атакующего, в общем случае, свои).

Вычислительная и теоретико-информационная маскировка

Определение вычислительной маскировки

Требование **вычислительной маскировки** (hiding) — распределения случайных величин $\mathrm{COMMIT}(U, 0)$ и $\mathrm{COMMIT}(U, 1)$ должны быть неразличимы **полиномиально ограниченным** атакующим E, когда случайная величина U принимает значения равновероятно из $\{0, 1\}^k$.

Атакующий может потратить на различение распределений порядка $p'(k)$ единиц времени и порядка $q'(k)$ единиц памяти (полиномы $p'(k)$ и $q'(k)$ для каждого атакующего, в общем случае, свои).

Определение теоретико-информационной маскировки

Если $\Delta(\mathrm{COMMIT}(U, 0), \mathrm{COMMIT}(U, 1))$ — это пренебрежимо малая функция по k (или равно нулю), то говорят о **теоретико-информационной маскировке**.

Вычислительная и теоретико-информационная маскировка

Определение вычислительной маскировки

Требование **вычислительной маскировки** (hiding) — распределения случайных величин $\text{COMMIT}(U, 0)$ и $\text{COMMIT}(U, 1)$ должны быть неразличимы **полиномиально ограниченным** атакующим E, когда случайная величина U принимает значения равновероятно из $\{0, 1\}^k$.

Атакующий может потратить на различение распределений порядка $p'(k)$ единиц времени и порядка $q'(k)$ единиц памяти (полиномы $p'(k)$ и $q'(k)$ для каждого атакующего, в общем случае, свои).

Определение теоретико-информационной маскировки

Если $\Delta(\text{COMMIT}(U, 0), \text{COMMIT}(U, 1))$ — это пренебрежимо малая функция по k (или равно нулю), то говорят о **теоретико-информационной маскировке**.

Еще одно определение вычислительной маскировки

Если $\Delta(E(\text{COMMIT}(U, 0)), E(\text{COMMIT}(U, 1)))$ — это пренебрежимо малая функция по k (или равно нулю).

О требованиях безопасности

Требования маскировки и связности покрывают только атаки со стороны отправителя или получателя. Но не защищают от вмешательства третьей стороны:

О требованиях безопасности

Требования маскировки и связности покрывают только атаки со стороны отправителя или получателя. Но не защищают от вмешательства третьей стороны:

- в фазе вручения обязательство $\mathbf{c} = \mathrm{COMMIT}(u, x)$ может быть подменено атакующим на $\mathbf{c}' = \mathrm{COMMIT}(u', x')$

О требованиях безопасности

Требования маскировки и связности покрывают только атаки со стороны отправителя или получателя. Но не защищают от вмешательства третьей стороны:

- в фазе вручения обязательство $\mathbf{c} = \mathrm{COMMIT}(u, x)$ может быть подменено атакующим на $\mathbf{c}' = \mathrm{COMMIT}(u', x')$
- в фазе раскрытия пара (u, x) может быть подменена парой (u', x').

О требованиях безопасности

Требования маскировки и связности покрывают только атаки со стороны отправителя или получателя. Но не защищают от вмешательства третьей стороны:

- в фазе вручения обязательство $\mathbf{c} = \mathrm{COMMIT}(u, x)$ может быть подменено атакующим на $\mathbf{c}' = \mathrm{COMMIT}(u', x')$
- в фазе раскрытия пара (u, x) может быть подменена парой (u', x'). Ни у отправителя, ни у получателя нет средств выявления атакующего типа **человек посередине**.

О требованиях безопасности

Требования маскировки и связности покрывают только атаки со стороны отправителя или получателя. Но не защищают от вмешательства третьей стороны:

- в фазе вручения обязательство $\mathbf{c} = \mathrm{COMMIT}(u, x)$ может быть подменено атакующим на $\mathbf{c}' = \mathrm{COMMIT}(u', x')$
- в фазе раскрытия пара (u, x) может быть подменена парой (u', x'). Ни у отправителя, ни у получателя нет средств выявления атакующего типа **человек посередине**.

Для защиты от таких атак нужно использовать аутентифицированные каналы (прежде, чем играть в «орла и решку» по телефону, нужно сначала аутентифицировать друг друга).

На основе хэш-функции

На основе хэш-функции

- На основе криптографической хэш-функции H: $\text{COMMIT}_0(u, x) = H(u \parallel x)$.

$$A \to B : \mathbf{c} = H(u \parallel x)$$
$$A \to B : u', x'$$
$$B : H(u', x') \stackrel{?}{=} \mathbf{c}$$

На основе хэш-функции

- На основе криптографической хэш-функции H: $\text{COMMIT}_0(u, x) = H(u \parallel x)$.

$$A \to B : \mathbf{c} = H(u \parallel x)$$
$$A \to B : u', x'$$
$$B : H(u', x') \stackrel{?}{=} \mathbf{c}$$

  - Свойство **однонаправленности** (preimage resistance) хэш-функции (см. Лекцию 3) обеспечивает (как минимум) вычислительную маскировку

На основе хэш-функции

- На основе криптографической хэш-функции H: $\text{COMMIT}_0(u, x) = H(u \parallel x)$.

$$A \to B : \mathbf{c} = H(u \parallel x)$$
$$A \to B : u', x'$$
$$B : H(u', x') \stackrel{?}{=} \mathbf{c}$$

  - Свойство **однонаправленности** (preimage resistance) хэш-функции (см. Лекцию 3) обеспечивает (как минимум) вычислительную маскировку
  - **Сильная устойчивость к коллизиям** (collision resistance) (см. Лекцию 3) обеспечивает вычислительную связность: сложно подготовить такие u, x и $u', 1 - x$, что
$$H(u \parallel x) = H(u \parallel 1 - x).$$

На основе дискретного логарифма

На основе дискретного логарифма

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.

На основе дискретного логарифма

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.
- Правило вычисления обязательства:

$$\text{COMMIT}_1(u, x) = g^u h^x, u \in_R \mathbb{Z}_n.$$

На основе дискретного логарифма

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.
- Правило вычисления обязательства:

$$\text{COMMIT}_1(u, x) = g^u h^x, u \in_R \mathbb{Z}_n.$$

- **Вычислительная связность.** Схема док-ва: пусть (u, x) и $(u', 1 - x)$ – это коллизия, тогда

$$g^u h^x = g^{u'} h^{1-x} \Rightarrow \log_q h = \frac{u - u'}{1 - 2x}.$$

Противоречие!

На основе дискретного логарифма

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.
- Правило вычисления обязательства:

$$\text{COMMIT}_1(u, x) = g^u h^x, u \in_R \mathbb{Z}_n.$$

- **Вычислительная связность.** Схема док-ва: пусть (u, x) и $(u', 1 - x)$ – это коллизия, тогда

$$g^u h^x = g^{u'} h^{1-x} \Rightarrow \log_q h = \frac{u - u'}{1 - 2x}.$$

  Противоречие!
- **Теоретико-информационная маскировка.** Схема док-ва: случайные величины g^U и $g^U h$ имеют одинаковое распределение, когда U распределено случайно и равновероятно, поэтому $\Delta(g^U, g^U h) = 0$.

На основе дискретного логарифма 2

На основе дискретного логарифма 2

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.

На основе дискретного логарифма 2

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.
- Правило вычисления обязательства:

$$\text{COMMIT}_2(u, x) = (g^u, h^{u+x}), u \in_R \mathbb{Z}_n.$$

На основе дискретного логарифма 2

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.
- Правило вычисления обязательства:

$$\text{COMMIT}_2(u, x) = (g^u, h^{u+x}), u \in_R \mathbb{Z}_n.$$

- **Теоретико-информационная связность.** Схема док-ва: нет такой коллизии (u, x) и $(u', 1 - x)$, что $(g^u, h^{u+x}) = (g^{u'}, h^{u'+x})$.

На основе дискретного логарифма 2

- Пусть $\langle g \rangle$ — циклическая группа с порождающим элементом g, $n = |\langle g \rangle|$ – порядок этой группы (большое, желательно простое число), $h \in_R \langle g \rangle$ — случайно выбранный из группы элемент, h известен всем, но $\log_g h$ никто не знает.
- Правило вычисления обязательства:

$$\text{COMMIT}_2(u, x) = (g^u, h^{u+x}), u \in_R \mathbb{Z}_n.$$

- **Теоретико-информационная связность.** Схема док-ва: нет такой коллизии (u, x) и $(u', 1 - x)$, что $(g^u, h^{u+x}) = (g^{u'}, h^{u'+x})$.
- **Вычислительная маскировка.** Без доказательства.

Домашнее задание

- Пусть в схемах COMMIT_1 и COMMIT_2 значение x выбирается из $\mathbb{Z}_n$. Проанализируйте свойства безопасности в этом случае.
- Что произойдет, если в схемах COMMIT_1 и COMMIT_2 известно значение $\log_g h$?
- Пусть $x \in \langle g \rangle$, $\mathrm{COMMIT}(u, x) = g^u x$, $u \in_R \mathbb{Z}_n$. Проанализируйте свойства безопасности этой схемы.

Теорема

Не существует СВБ одновременно обладающей теоретико-информационной связностью и теоретико-информационной маскировкой.

Док-во: DESK

Заключение

Спасибо за внимание!