Лекция 3

Криптосистемы с открытым ключом, кодирующие один бит

(Конспект: И. Сергей)

3.1 Постановка задачи

Алиса (**A**) и Боб (**B**) обмениваются сообщениями через ненадежный канал, прослушиваемый Чарли (**C**). Боб отправляет сообщение Алисе. Хотелось бы, чтобы она его получила, а Чарли не понял, что там имелось в виду. Для этих целей Алиса генерирует ключ, отправляемый Бобу. Само собой, этот ключ известен Чарли. После этого Алиса использует другой ключ, чтобы раскодировать полученное от Боба сообщение. Чарли же таким ключом не обладает.

3.2 Основные определения

Поскольку всё на свете можно зашифровать по одному биту, для начала ограничимся случаем $m \in \{0, 1\}$.

Определение 3.1. Криптосистема с открытым кючом — это полиномиальный по времени детерминированный алгоритм G, которому на вход подают параметр надежности (в унарной системе) и строчку случайных битов. А он по этой паре однозначно порождает две булевых схемы: для кодирования и для раскодирования.

$$G:\ (1^n, r_g) \mapsto (e, d),$$

где e — открытый ключ, а d — секретный ключ. При этом, схемы e и d работают следующим образом. На вход подается сообщение $m \in \{0, 1\}$

после чего e порождает по нему код какой-то (очевидно, полиномиальной от длины исходного сообщения) длины c:

$$e(m, r_e) \in \{0,1\}^*.$$

Замечание 3.1. Кодирующая функция обязательно должна брать случайные биты. В противном случае ноль и единица кодируются всегда одним и тем же образом. О надежности протокола в этом случае говорить не приходится.

В свою очередь, d принимает на вход код и, возможно, некоторые случайные биты r_d. Для корректной работы системы естественным является условие:

$$d(e(m, r_e), r_d) = m,$$

но мы его не всегда будем требовать.

Замечание 3.2. Заметим, что определение не говорит ничего о надежности такой криптосистемы.

Система *δ-корректна* (δ-*PKCS*), если

$$\forall m \quad \Pr\{d(e(m, r_e), r_d) = m\} \geq \delta$$

Вероятность берется по всем случайным битам, принимаемыми нашими схемами на вход (в том числе по r_g).

Замечание 3.3. 1-PKCS (или просто *PKCS*) — система, которая кодирует и раскодирует без ошибкок.

Замечание 3.4. Если мы захотим кодировать сразу много битов, нам придётся модифицировать определение, поскольку схема e, как и любая другая схема, имеет фиксированное количество входов. Достаточно добавить алгоритмы E и D, которые могут пользоваться e и d соответственно; кодировать и раскодировать в этом случае будем этими алгоритмами.

Теперь мы хотим, чтобы Чарли не мог отличить код нуля от кода единицы.

Определение 3.2. Криптосистема (для кодирования одного бита) называется *надежной*, если

$$\forall C \; \forall p \quad \Pr\{C(e(m, r_e), 1^n, e) = m\} \leq \frac{1}{2} + \frac{1}{p(n)} \ ,$$

где C — вероятностный полиномиальный по времени алгоритм, p — некоторый многочлен. Параметр 1^n дается для того, чтобы разрешить взломщику работать время, полиномиальное от параметра надёжности, а вероятность берется по всему, в том числе и по m (равному 0 или 1 равновероятно).

Получается, что C решает задачу следующего рода. Мы даем ему какой-то код — либо нуля, либо единицы. В системе без ошибок эти множества не пересекаются, C должен их разделить (он должен хотя бы иногда давать правильный ответ на этих множествах, а как он ведёт себя за их пределами — безразлично). Каждое из этих множеств по отдельности принадлежит классу **NP**. В самом деле, в данном случае подсказкой являются случайные биты кодирующей схемы e. Имея их, мы сможем проверить правильность кода.

Важно, что наши некоторые строки не являются кодами ни 0, ни 1. Представим себе систему, множество кодов которой представляет собой все возможные двоичные строки. Это будет означать, что мы ограничили себя множествами из $\mathbf{NP} \cap \mathbf{co\text{-}NP}$. Более того, если мы захотим, чтобы задача, которую мы решаем, была **NP**-трудной, нам придётся примириться с $\mathbf{NP} = \mathbf{co\text{-}NP}$.

Замечание 3.5. Таким образом, задача раскодирования криптосистемы — это частный случай разделения двух непересекающихся **NP**-множеств (*disjoint* **NP** *pairs*). Необходимость разделять такие множества возникает не только в криптографии, но и в теории сложности доказательств, где для автоматизации доказательства требуется отделить множество (невыполнимых) формул, имеющих короткие доказательства, от множества (выполнимых) формул, не имеющих доказательств.

3.3 PKCS на основе tdpf

Замечание 3.6. Почему в качестве криптосистемы с открытой ключом нельзя было воспользоваться tdpf? Все дело в том, что определение *trapdoor function* ничего не говорит о том, насколько сложно найти конкретный бит кодируемого сообщения. В случае сообщения из одного бита (а остальные биты в строке можем взять произвольным образом) *trapdoor function* кодирует *всю* строку, и ее найти по коду трудно. Но при этом, то, что нам надо (скажем, первый бит), может находиться сравнительно легко.

Попытаемся построить надежную криптосистему на основе *trapdoor function* G с кодирующей схемой e. Генерируемая нашей криптосистемой

G схема e' будет работать следующим образом:

$$e'(m, r) = (e(r), B(r) \oplus m),$$

где r — наши случайные биты, а B — трудный бит для e. Дешифровщик d' действует очевидным образом, обращая $e(r)$, получая затем $B(r)$ и делая XOR со второй компонентой кода.

Теорема 3.1. *В принятых выше обозначениях, если G — tdpf, а B — ее трудный бит, то G' — надежная криптосистема.*

Доказательство. Пусть у нас есть взломщик, который ломает полученную криптосистему. Взломаем с его помощью изначальное tdpf, или, по крайней мере, его трудный бит.

Мы получили $e(r)$ и пытаемся угадать $B(r)$. Для этого берем случайную строку b, и пару $(e(r), b)$ даем нашему взломщику (заметим, что это код какого-то сообщения). Взломщик говорит нам некоторое m, и если он его действительно отгадал, то $m \oplus b$ — искомый трудный бит. Взглянув на определения tdpf и PKCS, замечаем что все вероятности взлома — такие, как нам надо, значит мы взломали трудный бит. А не должны были. □

Следствие 3.1. *∃ tdpf ⇒ ∃ надежная криптосистема (PKCS).*

Упражнение 3.1. Доказать в обратную сторону.

3.4 Криптосистема, полная в $\frac{2}{3}$–PKCS

Вспомним определение сведéния между owf из предыдущей лекции и творчески его доработаем для криптосистем, поскольку надёжность у нас теперь сильная, а значит, взлом — слабый.

Определение 3.3. Алгоритм A взламывает криптосистему G с вероятностью $q(n)$, если для бесконечной последовательности длин n_i

$$\Pr_{b \in \{0,1\}, A, r_g, r_e} \{A(e(b, r_e), 1^{n_i}, e)) = b\} \geq q(n_i).$$

Определение 3.4. $F \to G$ (взлом криптосистемы F сводится к взлому криптосистемы G), если

$$\exists T^{\cdot}\ \forall p\ \exists p' :$$

$$A \text{ взламывает } G \text{ с вероятностью } \frac{1}{2} + \frac{1}{p(n)} \Rightarrow$$

$$T^A \text{ взламывает } F \text{ с вероятностью } \frac{1}{2} + \frac{1}{p'(n)}.$$

Здесь T — полиномиальная по времени оракульная машина Тьюринга, A используется как *вероятностный* оракул, $p(n)$ и $p'(n)$ — многочлены (положительные при $n \geq 1$).

Итак, будем строить полную относительно этого сведения криптосистему G в классе криптосистем, которые расшифровывают зашифрованное сообщение правильно с вероятностью $\frac{2}{3}$ ($\frac{2}{3}$–PKCS). Наша криптосистема будет пользоваться всеми на свете криптосистемами. Для параметра надежности n' можно воспользоваться первыми n' криптосистемами. Кодирование будет производиться так, что, если хотя бы одна из рассмотренных криптосистем является надежной, то и наша криптосистема надежна. Построение происходит в три этапа.

1. ***Сертификация.*** Для начала производится перебор алгоритмов и сертификация их в качестве криптосистем. В самом деле, нам подходит не любой алгоритм из перебираемых: он должен порождать пары схем, обратных друг к другу (с хорошей вероятностью). Нам нужно отобрать первые n' подходящих алгоритмов. Для этого мы производим проверку каждого из «кандидатов».

   Замечание 3.7. Мы перебираем алгоритмы, работающие некоторое, ограниченное конкретным полиномом (потом покажем, что этого достаточно), время.

   Сертификация проходит следующим образом. Для каждого «кандидата» G_i мы много раз производим шифрование и дешифрование случайных сообщений (однобитовых), выбирая различные случайные ключи. Если в подавляющем большинстве случаев получился правильный результат получился, добавляем нашего «кандидата» в список. Для прохождения сертификации мы будет требовать чуть более $\frac{2}{3}$ правильных ответов.

2. ***Amplification of correctness.*** Сделаем ошибку нашей криптосистем экспонициально малой. Для это много раз закодируем сообщение, а при раскодировании ответ возьмем по большинству. Следует проследить за тем, чтобы эта конструкция не уменьшила надежности.

3. ***Объединение.*** Объединим все выбранные криптосистемы таким образом, что, если среди них была хоть одна надежная, то и полученная криптосистема будет надежной. Для этого кодируемый бит

b представим в виде $b = b_1 \oplus \ldots \oplus b_{n'}$, где $b_1 \ldots b_{n'-1}$ — некоторые случайные биты, а $b_{n'} = b_1 \oplus \ldots \oplus b_{n'-1} \oplus b$. После этого каждый из битов кодируется своей криптосистемой. "Понятно, что" для для раскодирования взломщику нужно взломать все n' криптосистем.

Покажем, что для взлома любой криптосистемы достаточно взломать только что нами построенную. Для взлома конкретной криптосистемы G_*, которой закодировали некоторый бит b_*, впишем его код на место соответствующего случайного бита, закодированного G_* в нашей криптосистеме. (Для этого n' нужно взять достаточно большим: больше номера криптосистемы G_*.) Таким образом, по полученному значению бита b, зная все остальные случайные биты, мы получим и исходный бит b_*.

Замечание 3.8. Будем считать, что $G_1 = \mathrm{id}$. Тем самым построенная криптосистема будет правильно кодировать сообщения даже если не сертифицирована ни одна из перебираемых криптосистем.

Теорема 3.2. *Построенная криптосистема G — полная в классе $\frac{2}{3}$–PKCS.*

Определение 3.5. *Возведение криптосистемы в степень.* Для криптосистемы H определим $H^k = (e^k, d^k)$, где

$$e^k(m,r) = (e(m,r_1), e(m,r_2), \ldots, e(m,r_k)),$$

$$d^k = \mathrm{maj}_i \,\{d(e(m,r_i))\}.$$

Лемма 3.1. *$H \in \frac{2}{3}$–PKCS $\Rightarrow$*

1. *$H^k \in (1 - e^{-k\cdot\mathrm{const}})$–PKCS.*

2. *$H^k \to H$ (точнее, существует такое оракульное сведение R, что если некий «соперник» A ломает H^k с вероятностью $\frac{1}{2} + \frac{1}{p(n)}$, то R^A ломает H с вероятностью $\frac{1}{2} + \frac{1}{p(n)\cdot k}$).*

Доказательство. 1. Пусть X_i — случайная величина, соответствующая факту $d(e(m,r_i)) = m$. Применяя неравенство Чернова, получим

$$\Pr\left\{\sum X_i \le k/2\right\} = \Pr\left\{\sum X_i \le 2k/3 - k/6\right\} < e^{-\mathrm{const}\cdot k},$$

2. Будем строить такое сведение: пусть R^A выбирает случайным образом $i \in \{1 \ldots k\}$ и формирует свой запрос к A следующим образом:

$$M = \underbrace{e(0), e(0), \ldots, e(0)}_{i-1}, c, \underbrace{e(1), \ldots, e(1)}_{k-i},$$

где c — тот код, который мы хотим взломать. Этот вход мы даем R^A. Следует заметить, что такой вход может не являться корректным, так как в определении H^k предполагается, что на вход взломщику дается массив из кодов нулей либо единиц. Тем не менее, оценим вероятность успеха R^A.

$$\Pr\{\text{успеха } R^A\} = \frac{1}{2}\Pr\{\text{успеха} \mid c = e(0)\} + \frac{1}{2}\Pr\{\text{успеха} \mid c = e(1)\} =$$

$$\frac{1}{2}\Big(\frac{1}{k}\Pr\{R^A \text{ выдает } 0 \mid M = 0\ldots 0\} + \frac{1}{k}\Pr\{R^A \text{ выдает } 0 \mid M = 0\ldots 01\} + \ldots$$

$$\ldots + \frac{1}{k}\Pr\{R^A \text{ выдает } 1 \mid M = 1\ldots 1\} + \frac{1}{k}\Pr\{R^A \text{ выдает } 1 \mid M = 0\ldots 01\} + \ldots\Big) =$$

$$\frac{1}{2}\left(1 - \frac{1}{k}\right) + \frac{1}{2k}(\Pr\{R^A \text{ выдает } 0 \mid M = 0\ldots 0\} + \Pr\{R^A \text{ выдает } 1 \mid M = 1\ldots 1\}) \geq$$

$$\frac{1}{2}\left(1 - \frac{1}{k}\right) + \frac{1}{k}\left(\frac{1}{2} + \frac{1}{p(n)}\right) = \frac{1}{2} + \frac{1}{p(n)\cdot k}$$

□

Доказательство теоремы. Результат леммы дает нам экспоненциально малую вероятность ошибки построенной криптосистемы G. Даже будучи просуммированной n' раз, ошибка остается экспоненциально малой.

Покажем, что достаточно ограничиться криптосистемами, использующими время n^2. Предположим, что у нас есть более медленная криптосистема G_s. Сделаем из нее быструю криптосистему G_f, которая отличается от G_s лишь параметром надежности.

$$G_f(1^n, \ldots) = G_s(1^{\lfloor n^{1/k} \rfloor}, \ldots)$$

Корректность данной системы не изменилась, а надежность все так же оценивается полиномом.

Докажем полноту G: сведем взлом некоторой криптосистемы G^* ко взлому G. Разумеется, в этом случае n' больше, чем номер G^*, иначе она просто не будет рассмотрена среди составляющих G. Согласно лемме 3.1, мы сводим взлом G^* к взлому G^{*k}, а затем[1] взлом G^{*k} — ко взлому G. □

Упражнение 3.2. Определим tdpf с ошибкой (δ-tdpf). Предлагается построить полное δ-tdpf.

[1]Это надо уметь доказать формально!

Упражнение 3.3. До этого у нас «соперник» был вероятностный. Пусть теперь он будет детерминированным. Предлагается построить криптосистему, полную в классе криптосистем с детерминированным «соперником».

Замечание 3.9. Решение предполагает построение детерминированного сведения.