Доказательства с неразглашением II

Сергей Николенко

Криптография — CS Club, осень 2009

Outline

Graph-ISO

- Вспомним наш протокол для изоморфизма графов.
  1. P случайно выбирает перестановку π и передаёт V граф $C = \pi(H)$.
  2. V подбрасывает монетку, выбирает G или H и спрашивает его у P.
  3. Если P выбрал G, V передаёт $\sigma = \pi$, если V выбрал H, P передаёт $\sigma = \pi \circ \varphi$.
  4. V проверяет, что $\sigma($выбранного графа$) = C$.
- Удовлетворяет ли он нашему определению?

Graph-ISO

- Вспомним наш протокол для изоморфизма графов.
  1. P случайно выбирает перестановку π и передаёт V граф $C = \pi(H)$.
  2. V подбрасывает монетку, выбирает G или H и спрашивает его у P.
  3. Если P выбрал G, V передаёт $\sigma = \pi$, если V выбрал H, P передаёт $\sigma = \pi \circ \varphi$.
  4. V проверяет, что σ(выбранного графа) $= C$.
- Удовлетворяет ли он нашему определению?
- Да; симулятор, о котором мы говорили, может относиться к V как к чёрному ящику (т.е. в том числе как к чёрному ящику с подсказкой).

Протокол для NISO

- Рассмотрим нашу интерактивную систему доказательств для NISO.
  1. Вы случайно выбираете G или H и перестановку π.
  2. Посылаете мне результат применения π к выбранному графу.
  3. А я должен угадать, какой это был граф.
- Это система доказательств с неразглашением?

Протокол для NISO

- Если в этом протоколе V хочет решить задачу изоморфизма между графом C и одним из графов G и H, он может просто задать её пруверу, и P вынужден будет её решить.
- То есть V может вытащить из P некоторую информацию, которую сам подсчитать не может (потому что мы предполагаем, что V сам не сможет решить задачу изоморфизма графов).
- Наше определение тут, к счастью, ломается: симулятор не сможет восстановить ответ P на вопрос этого V.
- Почему оно сломалось? Что «философски» не так с этим протоколом?

Протокол для NISO

- С этим протоколом неладны две вещи.
  1. P — это детерминированный алгоритм (и поэтому симулятор не может рассчитывать его эмулировать за своё полиномиальное время).
  2. V может задавать P нетривиальные вопросы (и поэтому симулятор не может рассчитывать эмулировать возможные сообщения потенциально вредоносного V).
- Как его исправить так, чтобы избавиться от этих недостатков?

Подсказка

- Подсказка: чтобы V не мог задавать P нетривиальные вопросы, на самом деле хочется, чтобы:
  - *сначала* V доказал P, что знает, какому графу изоморфен C *на самом деле*;
  - а уже *потом* P сообщил V, какому графу C изоморфен *по мнению* P.

Протокол

- Вот новый протокол для NISO:
  1. V подбрасывает монетку и выбирает G или H (пусть G). V выбирает случайную перестановку π, вычисляет $X = \pi(G)$. Затем он бросает ещё $2k$ монеток, выбирает ещё $2k$ случайных перестановок $\pi_1, \ldots, \pi_{2k}$ и строит $X_1, X_2, \ldots, X_{2k}$. Затем V отсылает $X, X_1, \ldots, X_{2k}$.
  2. P подбрасывает $2k$ монеток и отправляет их V.
  3. V отправляет ещё одно сообщение:
     - если i-й бит сообщения P равен 0, V отсылает обратно π_i;
     - если i-й бит сообщения P равен 1, V либо отсылает обратно $\varphi_i : \varphi_i(X) = X_i$, если может (т.е. если X_i и X были сделаны из одного графа), либо, если не может, отсылает $\perp$ («не получилось»).
  4. P проверяет π_i и φ_i, а также проверяет, что $\perp$ не больше $\frac{k}{3}$. Если всё OK, отсылает V один бит, указывающий, $X \equiv G$ или $X \equiv H$.

Доказательство

- Итак, нам нужно теперь доказать, что этот протокол — действительно система доказательств с неразглашением.
- Иначе говоря, нужно доказать три свойства:

полнота: $\forall x \in L \Pr[(P, V)(x) = \text{“Да”}] \geq 1 - \epsilon(|x|)$;
корректность: $\forall x \notin L \, \forall P' \Pr[(P, V)(x) = \text{“Да”}] \leq \frac{1}{2}$;
неразглашение: $\forall V' \exists S \, \forall x \in L \, \forall a \, \mathrm{VIEW}_{P, V'(a)}(x) = S(x, a)$.

Полнота

- Почему $\forall x \in L \ \Pr[(P, V)(x) = \text{“Да”}] \geq 1 - \epsilon(|x|)$?

Полнота

- Почему $\forall x \in L \ \Pr[(P, V)(x) = \text{“Да”}] \geq 1 - \epsilon(|x|)$?
- Если $x \in L$, т.е. $G \not\equiv H$, P всегда может точно определить, что из двух: $C \equiv G$ или $C \equiv H$, и ответить правильно.
- Плохой случай — когда V честный, а P кажется, что V нечестный.
- Честный V может вернуть φ_i для каждого графа с вероятностью $\frac{1}{2}$, а плохой случай наступает, если ему не повезёт $\frac{2k}{3}$ раз из k.
- Вероятность этого экспоненциально (по k) мала (оценки Чернова — см. курс Димы).

Корректность

- Почему $\forall x \notin L \ \forall P' \ \Pr[(P, V)(x) = \text{“Да”}] \leq \frac{1}{2}$?

Корректность

- Почему $\forall x \notin L\ \forall P'\ \Pr[(P, V)(x) = \text{“Да”}] \leq \frac{1}{2}$?
- Тут всё просто. Если $G \equiv H$, то *все* участвующие в процессе графы изоморфны друг другу. Прувер это видит, но ничего поделать не может.
- И какие бы биты прувер не спрашивал, он не получит никакой информации; он может, получив $\perp$, узнать, что “базовый граф” X — это не “базовый граф” X_i, но какие это графы — нет никакой возможности узнать, они все изоморфны.

Неразглашение

- Почему $\forall V' \; \exists S \; \forall x \in L \; \forall a \; \mathrm{VIEW}_{P,V'(a)}(x) = S(x,a)$?
- Мы докажем даже более сильное утверждение:

$$\exists S \; \forall V' \; \forall x \in L \; \forall a \; \mathrm{VIEW}_{P,V'(a)}(x) = S(x,a).$$

- Иначе говоря, докажем, что симулятор сможет построить протокол общения с любым V', используя его как чёрный ящик.

Неразглашение

- Как построить S, который $\forall V' \; \forall x \in L \; \forall a \; \mathrm{VIEW}_{P,V'(a)}(x) = S(x,a)$?
  - S подбрасывает монетку и помещает то, что получается, на ленту случайных битов V'.
  - V' просыпается, создаёт $X, X_1, \ldots, X_{2k}$ (не обязательно по протоколу — он же зловредный V') и отсылает их S.
  - S подбрасывает $2k$ монеток, отсылает обратно V'. V' отвечает перестановками и $\perp$'ами.
  - S проверяет, что $\perp$'ов не слишком много и что все перестановки правильные. Если что-то не так, S прекращает разговор, считая, что V' его обманывает.
- Но если всё сошлось, S должен сказать V' ответ. Но ответа он не знает... что же делать?

Неразглашение

- S поступает в лучших традициях Чарли из некоторых наших атак.
- Он пока не отвечает V', а клонирует этот «чёрный ящик» в V'' и начинает новый разговор.
  - S даёт новому V'' *точно такие же случайные биты*.
  - V'' просыпается, создаёт $X, X_1, \ldots, X_{2k}$ — точно такие же, как у V', ведь случайные биты те же! — и отсылает их S.
  - S подбрасывает $2k$ (новых) монеток, отсылает обратно V''. V'' отвечает перестановками и $\perp$'ами.
  - S проверяет, что $\perp$'ов не слишком много и что все перестановки правильные. Если что-то не так, S прекращает разговор, считая, что V'' (а вместе с ним и V') его обманывает.

Неразглашение

- ...продолжение.
  - Если всё сошлось, S ищет позицию j среди этих $2k$, для которой один из V выдал π_j, а другой — φ_j (один отвечал на 0-запрос, другой — на 1-запрос, причём смог ответить).
  - Если такой есть, то S вычисляет базовый граф X как $\pi_j^{-1}(\varphi_j^{-1}(X))$ и отдаёт его V'.
  - Если такого нету, S начинает разговор с ещё одним клоном V'. Вероятность добиться успеха у S константная.

Неразглашение

- Осталось доказать, что S полиномиален и достигает успеха.
- К сожалению, «как написано» он полиномиален только в смысле «ожидаемое время работы полиномиально», потому что с маленькой вероятностью ему может не везти много раз подряд.
- Поэтому его придётся искусственно обрывать.
- И его распределение будет всё-таки немножко отличаться от настоящего прувера.
- Но отличаться незначительно; это мы уже формализовывать не будем.

Доказательства знания

- Однако обратите внимание на любопытный эффект: пока прувер доказывал, что $G \not\equiv H$, у V получилось доказать пруверу, что его X изоморфен то ли G, то ли H.
- И при этом V никак не раскрыл, какому именно графу изоморфен X.
- Это называется доказательством знания с неразглашением (zero-knowledge proof of knowledge).

Доказательства знания

- Пример: дискретный логарифм.
- Я показываю вам элемент $y \in G = \langle g \rangle$.
- Вы хотите убедиться, что я знаю, о чём говорю, т.е. знаю x, для которого $g^x = y$. А я не хочу этот x разглашать.
- Ситуация примерно как у Али-Бабы с купцом.
- Это тоже можно делать при помощи zero-knowledge.

Zero-knowledge для NP

- На самом деле можно построить интерактивные системы доказательства с неразглашением для всех языков из NP.
- Т.е. прувер может доказать, что формула выполнима, не предъявляя выполнимого набора и ничего о нём не сообщая.
- Здесь, правда, статистическая неразличимость распределений сменится уже вычислительной неразличимостью.
- А мы двинемся дальше, к другим применениям похожих идей.

Outline

Oblivious transfer

- Задача: есть сервер и клиент. У сервера есть набор строк $x_1, \ldots, x_n$, клиент хочет узнать x_i.
- Но при этом клиент не хочет, чтобы сервер узнал, какой бит ему нужен.
- Как её решить?

Oblivious transfer

- Задача: есть сервер и клиент. У сервера есть набор строк $x_1, \ldots, x_n$, клиент хочет узнать x_i.
- Но при этом клиент не хочет, чтобы сервер узнал, какой бит ему нужен.
- Как её решить?
- Тривиальное решение: сервер посылает клиенту всю строку $x_1 x_2 \ldots x_n$, клиент сам выбирает, что ему интересно.
- Но теперь сервер не хочет клиенту сообщать что-либо о других x_j, кроме запрошенного x_i.

Пример применения

- Пример применения: Алиса — это организатор «электронных денег». Она продаёт Бобу «электронный чек», по которому Боб что-нибудь купит, а потом продавец предъявит чек Алисе.
- Как это сделать?

Пример применения

- Пример применения: Алиса — это организатор «электронных денег». Она продаёт Бобу «электронный чек», по которому Боб что-нибудь купит, а потом продавец предъявит чек Алисе.
- Как это сделать?
- Тривиальное решение: Алиса готовит много сообщений вида «Я готова заплатить \$1» (плюс случайный паддинг), подписывает их электронной подписью и продаёт.
- Что тут не так?

Пример применения

- Понятно, что тут не так: Боб может скопировать сообщение и сделать себе магическим образом очень много «Алисиных денег». :)
- Как этого избежать?

Пример применения

- Понятно, что тут не так: Боб может скопировать сообщение и сделать себе магическим образом очень много «Алисиных денег». :)
- Как этого избежать?
- Алиса может раздавать много подписанных сообщений вида «Я готова заплатить \$1. ID=n».
- Тогда Алиса будет платить за каждый проданный чек только один раз и публиковать ID чеков, которые уже использованы.
- Что тут не так?

Пример применения

- Теперь это не устраивает Боба. Он, вообще говоря, пользовался этими чеками исключительно анонимности ради.
- А теперь Алиса всё знает про его покупки через её чеки.
- Вот для того, чтобы устроить всё ко всеобщему удовольствию, и нужен oblivious transfer.

Oblivious transfer Рабина

- Первая схема — OT Рабина.
- Суть: Алиса посылает бит через «зашумленный канал», после чего с вероятностью $\frac{1}{2}$ бит доходит до Боба, а с вероятностью $\frac{1}{2}$ — не доходит.
- Пока не очень понятно, как это связано с виртуальными чеками.

1-2-OT

- Вторая схема — 1-2-OT; Even, Goldreich, Lempel.
- Суть: Алиса посылает два бита (b_0, b_1) в «OT-машину». Боб посылает бит i, после чего «OT-машина» отдаёт Бобу бит b_i, а бит b_{1-i} выбрасывает.
- Это уже понятнее: Алиса готовит два чека вида «Я готова заплатить \$1. ID=i», но потом не знает, какой из них Боб выбрал.
- Правда, тут с электронными чеками есть маленький подвох, на который мы сейчас закроем глаза; какой?

Эквивалентность OT-протоколов

- Оказывается, эти два устройства эквивалентны.
- Если есть 1-2-OT, можно эмулировать зашумленный канал так:
  - Алиса хочет передать Бобу b с вероятностью $\frac{1}{2}$.
  - Она подбрасывает две монетки r и l и вводит в 1-2-OT-машину биты $b_l := b$ и $b_{1-l} := r$.
  - Боб запрашивает у машины бит i (i он может выбирать как угодно).
  - *После этого* Алиса открытым текстом посылает Бобу l.
- После этого Боб знает, что если $i = l$, то у него искомый бит b, а если $i \neq l$, то у него случайный мусорный бит r. А Алиса не знает, что Боб спрашивал у OT-машины.

Эквивалентность OT-протоколов

- Если есть зашумленный канал, можно эмулировать 1-2-OT так:
  - Алиса хочет передать Бобу b_0 или b_1 по выбору Боба так, чтобы Алиса не знала об этом выборе.
  - Она передаёт через зашумленный канал много случайных битов (скажем, $3n$) $s_1 s_2 \dots s_{3n}$.
  - Боб посылает Алисе два множества индексов I_0, I_1 с такими свойствами:
    - $|I_0| = |I_1| = n$;
    - одно из множеств, I_i (где i — это то i, которое Боб хочет узнать), состоит только из индексов, для которых Боб получил биты, а не $\perp$;
    - Другое множество I_{1-i} выбирается случайно.
  - Алиса посылает Бобу $b_0 \bigoplus_{j \in I_0} s_j$ и $b_1 \bigoplus_{j \in I_1} s_j$
- После этого Боб с большой вероятностью знает тот бит, который надо, и не знает того, которого не надо.

Замечание

- Важное замечание: если Боб — нечестный Боб, и подаст Алисе и I_0, и I_1 из «области полной информации», Алиса этого никак не сможет заметить.
- Чтобы это исправить, нужно, чтобы Боб мог сгенерировать случайный бит так, чтобы Алиса его не знала, но поверила, что он случайный.
- Об этом мы поговорим чуть попозже.

Секретное вычисление функции

- Scrooge McDuck и Flintheart Glomgold хотят выяснить, у кого больше денег.
- Но на этот раз, памятуя об истории с 10-центовой монеткой, Скрудж не хочет физически взвешивать деньги.
- И вообще не хочет, чтобы Флинтхарт узнал, сколько денег у Скруджа.
- Им нужно вычислить функцию $f(a, b) = [a > b]$, причём один из них знает a, другой знает b, и никто не хочет разглашать информацию о своей части входа другому.

Секретное вычисление функции

- Основная идея: использовать вычисление функции при помощи *булевой схемы*.
- Рассмотрим схему, вычисляющую бит $[a > b]$.
- Скрудж знает половину входов этой схемы $\mathbf{a}$, Флинтхарт — другую половину $\mathbf{b}$.
- Им нужно обоим узнать результат, не сообщив друг другу ничего больше.

Секретное вычисление функции

- Инициализация:
  - Скрудж (S) генерирует случайную строку $\mathbf{a}^F$, вычисляет $\mathbf{a}^S = \mathbf{a} \oplus \mathbf{a}^F$ и посылает её Флинтхарту.
  - Флинтхарт (F) генерирует случайную строку $\mathbf{b}^S$, вычисляет $\mathbf{b}^F = \mathbf{b} \oplus \mathbf{b}^S$ и посылает её Скруджу.
- Теперь им нужно провести все вычисления по схеме. Они будут поддерживать инвариант: для каждого значения x в схеме F и S знают части x^F и x^S, такие, что $x = x^F \oplus x^S$. В начале это верно. Если удастся так дойти до конца, они смогут потом просто послать друг другу свои части ответа, и каждый будет знать только ответ и то, что из ответа можно вычислить, но ничего больше.

Секретное вычисление функции

- Пусть на входе гейта x и y, Скрудж знает x^S и y^S, Флинтхарт знает x^F и y^F, причём $x = x^F \oplus x^S$, $y = y^F \oplus y^S$. Хотят вычислить результат гейта.
- XOR-гейт $x \oplus y$: для него всё просто.
  - Скрудж (S) вычисляет $x^S \oplus y^S$.
  - Флинтхарт (F) вычисляет $x^F \oplus y^F$.
- И всё.

Секретное вычисление функции

- AND-гейт $x \wedge y$: тут сложнее. Они хотят вычислить

$$x \wedge y = \left(x^F \oplus x^S\right) \wedge \left(y^F \oplus y^S\right) =$$
$$= \left(x^S \oplus y^S\right) \wedge \left(x^F \oplus y^S\right) \wedge \left(x^S \oplus y^F\right) \wedge \left(x^F \oplus y^F\right).$$

- Вычислим сначала $x^S \wedge y^F$.
  - Скрудж (S) генерирует случайный бит r^S и подаёт на 1-2-OT-машину пару $\left(\left(x^S \wedge 0\right) \oplus r^S, \left(x^S \wedge 1\right) \oplus r^S\right)$.
  - Флинтхарт (F) подаёт в машину y^F.
  - Машина выдаёт Флинтхарту $\left(x^S \wedge y^F\right) \oplus r^S$.
- Теперь новая доля Скруджа $\left(x^S \wedge y^F\right)^S = r^S$, новая доля Флинтхарта $\left(x^S \wedge y^F\right)^F = \left(x^S \wedge y^F\right) \oplus r^S$, и никакой информации.

Секретное вычисление функции

- Точно так же получатся новые доли $\left(x^F \wedge y^S\right)^F = r^F$ и $\left(x^F \wedge y^S\right)^S = \left(x^F \wedge y^S\right) \oplus r^F$.
- И можно будет собрать $x \wedge y$:

$$(x \wedge y)^S = (x^S \wedge y^S) \oplus r^S \oplus \left(x^F \wedge y^S\right)^S,$$

$$(x \wedge y)^F = (x^F \wedge y^F) \oplus r^F \oplus \left(x^S \wedge y^F\right)^F.$$

- Так они дойдут до конца схемы, а потом перешлют друг другу свои части последнего значения.

Outline

Бросание монетки в колодец

- Мы хотим, чтобы Боб кинул монетку так, чтобы Алиса была уверена, что Боб кинул её честно, но чтобы сама Алиса не могла видеть результат этой монетки.
- Схема такая: Боб стоит у колодца, Алиса издалека кидает монетку в колодец, Боб туда заглядывает, а Алису не подпускает.
- Конечно, тут Боб может просто соврать о том, что в колодце; с этим мы потом разберёмся.

Бросание монетки в колодец

- Bit commitment: протокол, при котором Алиса посылает Бобу нечто, что Боб не может расшифровать, но что Алиса не может подделать.
- Например, на основе дискретного логарифма: секрет Алисы — x, Алиса посылает Бобу g и g^x.
- Теперь Боб сам не узнает x, но если Алиса соврёт, что у неё x другой, Боб её сможет проверить.
- Будем считать, что задана некоторая схема bit commitment c.

Бросание монетки в колодец

- Итак, алгоритм бросания монетки в колодец:
  - Боб подбрасывает монетку r_1.
  - Боб посылает Алисе $c(r_1)$.
  - Алиса посылает Бобу случайный бит r_2.
  - Боб использует как монетку результат $r_1 \oplus r_2$.
- Но тут Боб всё равно может соврать, он ведь Алисе потом посылает не сам r, а результат какого-то вычисления $f(y, r)$, где y — данные Боба, а Алиса не может сама расшифровать r.
- Поэтому нужно Алисе как-то убедиться в том, что Боб не врёт.

Бросание монетки в колодец

- Боб должен убедить Алису в том, что его сообщение m действительно соответствует $f(y, r_1 \oplus r_2)$ для некоторого r_1, для которого $c(r_1)$ совпадает с тем, что получила Алиса.
- Боб у нас, предположим, полиномиальный, т.е. f — полиномиально вычислимая функция.
- Давайте запишем такое утверждение:

$$\exists r' \left(m = f(y, r' \oplus r_2) \wedge c(r') = c(r_1) \right).$$

- Может ли Алиса его как-нибудь проверить?

Бросание монетки в колодец

- Утверждение:

$$\exists r' \left(m = f(y, r' \oplus r_2) \wedge c(r') = c(r_1) \right).$$

- Множество y и r_1, для которых это верно, лежит в классе NP, потому что, зная y и r', оба утверждения легко проверить.
- Но любое NP-утверждение мы можем доказать с неразглашением!
- Значит, алгоритм такой: Алиса и Боб сначала запускают алгоритм бросания монетки, а потом Боб убеждает Алису, не раскрывая r_1 и y, что он всё сделал честно.
- Это, конечно, жутко неэффективно, но наша задача сейчас — понять идею.

Покер по телефону

- Пример: несколько компьютеров решили сыграть в покер.
- Как честно раздать колоду, если все участвуют в игре?
- Раздать колоду — это значит выдать Алисе некоторую $f(T, \mathbf{r})$, где $\mathbf{r} = \mathbf{r}^A \oplus \mathbf{r}^B$ (случайные биты берут у всех, чтобы они были действительно случайными) так, чтобы Боб ничего не узнал о $f(T, \mathbf{r})$, и Алиса с Бобом не узнали случайных битов друг друга.

Покер по телефону

- Алгоритм покера по телефону:
  - Алиса бросает Бобу в колодец случайную строку $\mathbf{r}^B$;
  - Боб бросает Алисе в колодец случайную строку $\mathbf{r}^A$;
  - Алиса и Боб вычисляют свои доли $f(T, \mathbf{r})^A$ и $f(T, \mathbf{r})^B$ алгоритмом секретного вычисления функции.
  - Боб посылает Алисе $f(T, \mathbf{r})^B$ (а Алиса Бобу свою часть не посылает).

Спасибо за внимание!

- Lecture notes и слайды будут появляться на моей homepage:

  `http://logic.pdmi.ras.ru/~sergey/`

- Присылайте любые замечания, решения упражнений, новые численные примеры и прочее по адресам: `sergey@logic.pdmi.ras.ru`, `snikolenko@gmail.com`
- Заходите в ЖЖ  **smartnik**.