Лекция 7

Совместное вычисление функции

(Конспект: И. Монахов)

КОНСПЕКТ НЕ ПРОВЕРЯЛСЯ ЛЕКТОРОМ

7.1 Определения

Алиса A и Боб B имеют входы x,y соответственно и случайные биты, к тому же они могут посылать друг другу какие-нибудь сообщения(некоторые функции от уже полученных сообщений, и локальной информации). В результате каждая из сторон получает некоторую совместно вычисленную функцию $f(x, y)$, которую локально выдает.

Но после получения этой функции они могут попытаться узнать что-нибудь еще по накопленной при исполнении протокола информации.

Определение 7.1. Протокол называется надежным, если ни одна из сторон не может узнать ничего кроме своего ответа. Более формально:

Для любой Алисы A и Боба B можно по ним построить такие A' и B', что

$$A'(x, f(x, y)) \equiv A$$

$$B'(y, f(x, y)) \equiv B$$

Замечание 7.1. Это определение надежности для semi-honest (почти честных) Алисы и Боба: они четко следуют протоколу, за исключением того, что запоминают промежуточные данные и после завершения протокола могут продолжить что-то вычислять. Бывают еще malicious (зловредные), которые только делают вид, что играют по протоколу.

Замечание 7.2. Есть способ свести протоколы для почти честных противников к протоколам для зловредных. Можно построить компилятор, который будет преобразовывать одни в другие, но у нас этой конструкции не будет, потому что для нее нужно знать zero-knowledge proofs.

7.2 Yao протокол

Итак, мы хотим надежно вычислить функцию. Можно считать, что функция представлена в виде схемы. Построим по ней некоторую запутанную схему и перешлем ее другой стороне.

В этой схеме по ребрам будут бегать не нолики и единички, а какие-то неведомые зверушки. А именно, мы сопоставим битам некоторые случайные строчки w^0,w^1 длины n.

Но как вычислять гейт от строчки? Для этого будет специальная табличка (w^i, u^j, v^k), такая что $k = i \circ j$.

Конечно, нельзя просто так передавать третий столбец таблички, потому что тогда враг может догадаться, какой это на самом деле был гейт и какие биты пришли ему на вход. Вместо этого будем передавать столбец $E_{w^i}(E_{u^j}(v^k))$, где E - это алгоритм кодирования некоторой криптосистемы с закрытым ключом.

Замечание 7.3. Нужно еще, чтобы для такой криптосистемы был способ проверить, павильно ли мы раскодировали слово. Этого можно добиться, например, при помощи контрольных сумм. Можно еще потребовать, чтобы у криптосистемы были дизъюнктные множества кодов для разных ключей.

Замечание 7.4. Можно также построить похожий протокол с использованием криптосисттемы с открытым ключом, но это сложнее.

Выходной бит схемы получить не сложно - нужно просто прислать пары битов с сопоставленными им строчками.

Осталось запутать входы сторон. Входные биты стороны, делавшей схему, просто кодируются соответствующими строчками. Проблема со входом другой стороны. Нельзя просто прислать противнику обе строчки, потому что тогда он сможет вычислить функцию для любого своего входа, а это слишком много. Чтобы завершить протокол нужно использовать конструкцию Oblivious Transfer для передачи входных строчек второй стороны.

7.3 Oblivious Transfer

У Алисы(Sender) есть две строчки $b_0, b_1 \in \{0,1\}^n$, а у Боба(Receiver) есть бит $i \in \{0,1\}$. После завершения протокола Receiver получает b_i, а Sender получает шиш.

Утверждение 7.1. *В предположении, что существует enhanced tdpf можно построить надежный протокол.*

Замечание 7.5. На самом деле, достаточно построить протокол для строчек длины 1, а потом запустить его параллельно.

1. Sender генерирует (e, d) и посылает e Receiver'у.
2. Receiver вычисляет $a_i = e(r_1)$ и $a_{i-1} = s(r_2)$, где s - усиленный Sampler, который выдает случайную строчку из $\operatorname{Im} e$, неотличимую от образа обычной строчки. Потом Receiver посылает (a_0, a_1) Sender'у.
3. Sender вычисляет $c_j = b_j \oplus B(d(a_j))$, посылает Receiver'у (c_0, c_1), и локально выдает пустую строчку.
4. Receiver вычисляет $B(r_1) \oplus c_i = b_i$ и выдает b_i.