Введение

В данной работе мы рассматриваем односторонние функции, которые являются одним из основных криптографических примитивов. Криптография известна еще с древних времен, и на протяжении большей части истории ее развития при кодировании использовался так называемый "закрытый" ключ, с помощью которого можно было как кодировать сообщение, так и раскодировать его. Соответственно этот ключ нужно было хранить в секрете, и знать его должны были только те, между кем происходил обмен сообщениями. Однако в 1976 году двумя американскими исследователями Диффи и Хеллманом была предложена новая схема передачи информации, которая сейчас известна как криптография с открытым ключом. Основная идея довольно проста: для шифрования используется так называемая "односторонняя" функция. Односторонняя функция - функция, которую легко посчитать в одну сторону, но тяжело обратить, то есть для любого входа x из области определения f можно легко получить $f(x)$, однако тяжело получить значение $f^{-1}(y)$ для любого y из множества значений функции f. При этом существует так называемый "открытый" ключ, который используется отправителем информации для ее шифрования и известен всем, и "закрытый" ключ, который известен только получателю информации. Криптография с открытым ключом следует правилу, впервые сформулированному голландским криптографом Керкхоффом: стойкось шифра, то есть криптосистемы - набора процедур, управляемых некоторой секретной информацией небольшого объема, должна быть обеспечена в том случае, когда криптоаналитику противника известен весь механизм шифрования за исключением секретного ключа - информации, управляющей процессом криптографических преобразований.

Самая известная и используемая на практике криптографическая функция с открытым ключом была предложена Ривестом, Шамиром и Адлеманом, и в их честь она была названа RSA. Она играет очень важную и все возрастающую роль в передаче сообщений и обращении к базам данных через сеть. Однако главная проблема современной практической криптографии состоит в том, что ни для одной используемой функции с открытым ключом не удалось доказать ее односторонность.

Задачи, возникающие в теоретической информатике, естественным образом делятся на сложностные классы. Например, класс P состоит из задач, которые можно решить на детерминированной машине Тьюринга, заканчивающей работу за время, полиномиальное от длины ее входа, а класс NP - из задач, решаемых за полиномиальное время на недетерминированной машине Тьюринга (подробное определение машины Тьюринга дано в секции ...).

Развитие теории сложности вычислений началось с работ, исследующих сложность отдельных задач. Однако первые же исследования показали, что нужны методы, позволяющие исследовать сложностные классы в целом, так как рассматривать каждую задачу по отдельности нет никакой возможности. Для теоретической информатики крайне важны возможности,

появляющиеся при наличии в сложностном классе полной задачи, так как тогда можно сместить предмет анализа с класса в целом (о котором обычно "в лоб" мало что можно доказать) на одну конкретную, хорошо заданную задачу.

В криптографии о полных задачах долгое время вообще ничего не было известно. В то время как "обычные" сложностные классы обзавелись полными задачами относительно скоро, между определением криптосистемы с открытым ключом [1] и полной задачей для класса криптосистем с открытым ключом (с ошибкой) [2] прошло тридцать лет. Более того, разработанные полные криптосистемы пока что относятся к "плохой" разновидности полных задач: они требуют перечисления всех машин Тьюринга, и считалось, что они вряд ли могут иметь дальнейшее применение как для теоретического анализа сложности или надежности, так и для практического применения.

Вся современная теоретическая криптография постороена на предположении существования односторонней функции, однако это предположение не дает нам кода машины Тьюринга, реализующей эту одностороннюю функцию. Так что все усилия теоретической криптографии в конечном счете направлены на предъявление конкретной односторонней функции. В криптографии, как и во всей теоретической информатике, важную роль играют полные конструкции, так как эффективная обратимость полной односторонней функции ведет к полиномиальной обратимости всех полиномиально вычислимых функций, а значит к несуществованию односторонних. Однако в криптографии полные конструкции носили до сих пор исключительно теоретический характер, так как полные односторонние функции - это функции, которые становятся односторонними, если на часть входа, принимающую код машины Тьюринга, им подается код односторонней функции, которого пока что никто не знает.

Однако в этой работе показано, что полные односторонние функции также могут быть использованы на практике, причем для этого совершенно не нужно знать конкретного кода машины Тьюринга, которая бы реализовывала одностороннюю функцию, то есть фактически доказывается, что полная односторонняя функция, на вход которой подается строка, которая одновременно является и кодом машины Тьюринга, и ее входом - односторонняя в предположении существования односторонних функций. Также рассматривается возможность практического применения композиции алгоритма RSA с полной односторонней функцией, которая бы с одной стороны работала практически так же хорошо, как RSA, на входах небольшой длины, и была бы плохо обратима в предположении о существовании односторонних функций начиная с некоторой длины входа. Фактически мы предъявляем конкретную одностороннюю функцию, кажущуюся хорошо работающей на практике в предположении о существовании односторонних функций.

Общие определения

В этом параграфе мы введем определения, необходимые для дальнейшей работы, а также определение сложности в среднем, которое нужно для понимания стоящих перед нами задач.

Определение. *Детерминированная машина Тьюринга - это семерка* $M = \langle Q, \Gamma, B, \Sigma, \pi, s, H \rangle$, *где:*
1) Q *- конечное множество состояний машины Тьюринга;*
2) Γ *- конечный алфавит символов ленты;*
3) $B \in \Gamma$ *- пустой символ (единственный символ, который может встречаться на ленте бесконечное число раз);*
4) $\Sigma \subseteq \Gamma \backslash B$ *- алфавит символов подаваемых на вход;*
5) $\pi : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$ *- функция перехода, описывающая работу машины Тьюринга:* L *обозначает сдвиг головки влево,* R *- вправо,* N *- отсутствие сдвига;*
6) $s \in Q$ *- начальное состояние машины Тьюринга;*
7) $H \subseteq Q$ *- множество конечных состояний.*

Неформально говоря, машина Тьюринга состоит из бесконечной ленты и головки, расположенной над одной из ячеек этой ленты. На ленте в начальном состоянии s записан вход, являющийся строкой над Σ. Машина производит переходы из одного состояния в другое в соответствии с функцией π, на вход которой подается текущее состояние $q \in Q$ и символ $\alpha \in \Gamma$, который на данном шаге находится под головкой. Головка может двигаться влево или вправо, или оставаться на месте в зависимости от значения третьей компоненты результата функции π. Машина Тьюринга M вычисляет функцию $f : \Sigma^* \to \Gamma^* \backslash \{B\}$, если при запуске M со входом $x \in \Sigma^*$ она заканчивает работу (переходит в конечное состояние), оставляя на ленте строку $f(x)$. В дальнейшем мы для простоты не будем делать различий между алфавитами Σ и $\Gamma \backslash \{B\}$. Обозначим через $t_M(x)$, где $t_M : \Sigma^* \to \mathbb{N}$, количество шагов, которое требуется машине Тьюринга M, чтобы, получив на вход x, перейти в конечное состояние. Машина Тьюринга M работает за время $T : \mathbb{N} \to \mathbb{N}$, если $\forall n \in \mathbb{N} \quad \max\limits_{x:|x|=n} t_M(x) \leq T(n)$.

Определение. *Недетерминированная машина Тьюринга - это обычная детерминированная машина Тьюринга, вход которой состоит из двух частей: собственно входа и "подсказки". Недетерминированная машина Тьюринга* M *вычисляет функцию* $f : \Sigma^* \to \Sigma^*$, *если для каждого входа* x *существует такая подсказка* $y \in \Sigma^*$, *что* $M(x, y) = f(x)$.

Более неформальное определение недетерминированной машины Тьюринга состоит в том, что π может быть не только функцией, но и отношением, то есть для некоторых конфигураций машины Тьюринга существует несколько равноправных вариантов действий, и недетерминированная машина Тьюринга параллельно начинает исполнять все такие варианты.

Определение. *Вероятностная машина Тьюринга - это практически то же самое, что и недетерминированная, изменяется только интерпретация: символы "подсказки" теперь интерпретируются как случайные символы машины. Вероятностная машина Тьюринга вычисляет функцию f на входе x с вероятностью p, если*

$$\Pr_{y \in U_m} (M(x, y) = f(x)) = p$$

где m - количество случайных битов, а U_m - равномерное распределение на строках длины m из алфавита Σ. В дальнейшем обычно $\Sigma = \{0, 1\}$.

Далее для краткости слова "машина Тьюринга" мы будем заменять аббревиатурой "МТ".

Сейчас мы переходим к определению сложности в среднем. Несмотря на многие годы изысканий до сих пор не найдено эффективных алгоритмов, решающих NP - полные задачи, и поэтому они считаются трудно вычислимыми. Однако NP - полнота подразумевает время работы в худшем случае. NP - полнота задачи не дает информации о том, насколько тяжело задача решается в среднем. В самом деле показано, что некоторые NP - полные задачи хорошо решаются для "среднего" случая. Например, хотя задача существования Гамильтонова пути в графе NP - полна, Гуревич и Шелах [3] показали, что если граф берется согласно обычному распредлению на произвольных графах, то ожидаемое время работы алгоритма, решающего задачу существования Гамильтонова пути, - линейное. Таким образом сложность задачи в среднем во многих отношениях более интересна, нежели ее сложность в худшем случае. Задачей криптографии является предъявление функции, которая была бы тяжело взламываема в среднем, однако на некоторых входах она может быть легко обратимой.

Композиция полной односторонней функции с функцией, регулярной по длине

Необходимые определения

Теперь перейдем непосредственно к определению односторонних функций. Далее заглавной буквой A мы будем обозначать алгоритм, пытающийся взломать одностороннюю функцию, за p обозначим полином от длины строки, а саму длину обозначим за n. В свою очередь U_n - множество строк длины n с равномерным распределением на них. Маленькими буквами из конца латинского алфавита обозначены бинарные строки.

Определение. *Функция f называется сильной односторонней (strong one-way), если*

1. $\exists A$ - вероятностный, полиномиальный по времени, такой что $\forall x$ - входа $A(x) = f(x)$.

2. $\forall A'\ \forall p\ \exists N : \forall n > N \Pr\limits_{x \in U_n}(A'(f(x), 1^n) \in f^{-1}f(x)) < \dfrac{1}{p(n)}$.

Определение. *Функция f называется слабой односторонней (weak one-way), если*

1. $\exists A$ *- вероятностный, полиномиальный по времени, такой что* $\forall x$ *- входа* $A(x) = f(x)$.

2. $\exists p\ \forall A'\ \exists N : \forall n > N \Pr\limits_{x \in U_n}(A'(f(x), 1^n) \notin f^{-1}f(x)) > \dfrac{1}{p(n)}$

Определение. *Функция f называется регулярной по длине (length-regular), если* $\forall x, y \in \{0,1\}^* \quad |x| = |y| => |f(x)| = |f(y)|$.

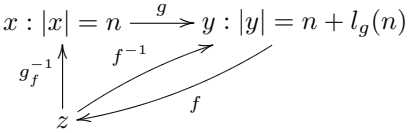
Определение. *Функция f называется сохраняющей длину (length-preserving), если* $\forall x \in \{0,1\}^* \quad |f(x)| = |x|$.

Доказательство свойств композиции

Теорема. *Пусть f - сильная односторонняя, g - полиномиально вычислимая, регулярная по длине, инъекция, увеличивает длину не больше чем на $O(\log_2 n)$, то есть пусть $l_g(n) := |g(x)| - |x|, |x| = n$, то $l_g(n) \leq O(\log_2 n)$. Тогда $f(g)$ - сильная односторонняя.*

Для доказательства нам потребуется следующая лемма:

Лемма. *Пусть f - произвольная, g - регулярная по длине, инъекция и $f_g = f(g) : \exists A\ \exists \alpha\ \forall N\ \exists n > N$: $\Pr\limits_{x \in U_n}(A(f_g(x), 1^n) \in f_g^{-1}f_g(x)) \geq \alpha(n)$, где α - функция от длины строки, которую можно выбирать.*
Тогда $\exists A'\ \forall N\ \exists n > N$: $\Pr\limits_{y \in U_{n+l_g(n)}}(A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)) \geq 2^{-l_g(n)} * \alpha(n)$, *где* $l_g(n) = |g(x)| - |x|, |x| = n$.



Доказательство. Так как g действует из $x : |x| = n$ в $y : |y| = n + l_g(n)$ и g - инъекция, то $\Pr\limits_{y \in U_{n+l_g(n)}}(y \in g(U_n)) = 2^{-l_g(n)}$, так как число строк длины $n + l_g(n)$ в $2^{l_g(n)}$ раз больше, чем длины n.

Построим алгоритм A' следующим образом:

1) A' получает на вход строку $z = f(y) : |y| = n + l_g(n)$ и $n + l_g(n)$ единичек.

2) A' применяет к z алгоритм A.

3) A' возвращает $g(x')$

Рассмотрим различные случаи для $y \in U_{n+l_g(n)}$. Нас интересует только нижняя граница вероятности того, что $A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)$.

1. Пусть $y \notin g(U_n)$. Тогда $\Pr\limits_{y \in U_{n+l_g(n)}, y \notin g(U_n)}(A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)) \geq 0$ и равенство достигается в случае, если f - инъекция, так как в этом случае $\forall y \in U_{n+l_g(n)} : y \notin g(U_n)\ f^{-1}f(y) = y$, но $A'(f(y), 1^{n+l_g(n)}) \in g(U_n)$ по построению A', а значит $A'(f(y), 1^{n+l_g(n)}) \notin f^{-1}f(y)$.

2. Пусть $y \in g(U_n)$. $\Pr\limits_{x \in U_n}(A(f_g(x), 1^n) \in f_g^{-1}f_g(x)) \geq \alpha(n)$.

Следовательно $\Pr\limits_{x \in U_n}(A(f_g(x), 1^n) \in f_g^{-1}f_g(x)) =$

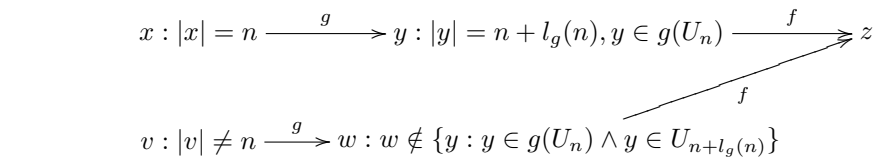
$\Pr\limits_{y \in U_{n+l_g(n)}, y \in g(U_n)}(A(f(y), 1^n) \in f_g^{-1}f(y)) =$

$\Pr\limits_{y \in U_{n+l_g(n)}, y \in g(U_n)}(A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)) \geq \alpha(n)$.

Первое равенство выполняется, так как g - инъекция и регулярная по длине, а тогда $\#(\{x : x \in U_n\}) = \#(\{y : y \in g(U_n) \wedge y \in U_{n+l_g(n)}\})$ следовательно вероятность, соответствующая отдельному x, переходит в вероятность, соответствующую $y = g(x)$.

Во втором равенстве используется то, что алгоритм A' использует в своей работе алгоритм A, а значит остается проверить только то, что если для соответствующего фиксированного y выполнено условие $A(f(y), 1^n) \in f_g^{-1}f(y)$, то выполнено и $A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)$. Если f - инъекция, то это очевидно верно. Если же нет, то возможен случай, когда алгоритм A успешно обратил $f_g(x)$, однако на выходе выдал строку, $x' : |x'| \neq n$. Эта возможность возникает, так как прообразу функции f могут принадлежать строки $w \notin \{y : y \in g(U_n) \wedge y \in U_{n+l_g(n)}\}$.

Эта ситуация показана на нижеследующей диаграмме:



Однако $\forall x \in f_g^{-1}f_g(U_n) \qquad g(x) \in f^{-1}f(g(U_n))$, а значит равенство выполняется, так как алгоритм A' на третьем шаге применяет к результату алгоритма A функцию g.

В итоге для $y \in U_{n+l_g(n)}$ получаем $\Pr\limits_{y \in U_{n+l_g(n)}}(A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)) \geq$

$\Pr\limits_{y\in U_{n+l_g(n)}}(y \in g(U_n)) \cdot \Pr\limits_{y\in U_{n+l_g(n)}, y\in g(U_n)}(A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)) \geq$
$2^{-l_g(n)} \cdot \alpha(n)$.

□

Доказательство теоремы. От противного: Пусть $f(g)$ - не сильная односторонняя. Тогда

$\exists A\ \exists p\ \forall N\ \exists n > N$: $\Pr\limits_{x\in U_n}(A(f_g(x), 1^n) \in f_g^{-1}f_g(x)) \geq \dfrac{1}{p(n)}$

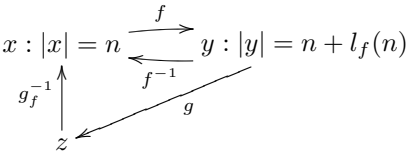
Возьмем $\alpha = \dfrac{1}{p(n)}$, и тогда по лемме $\exists A'\ \forall N\ \exists n > N$:

$\Pr\limits_{y\in U_{n+l_g(n)}}(A'(f(y), 1^{n+l_g(n)}) \in f^{-1}f(y)) \geq 2^{-c\cdot\log_2 n} \cdot \dfrac{1}{p(n)} = \dfrac{1}{n^c \cdot p(n)}$

Соответственно $\exists A'\ \exists p' = n^c \cdot p(n)\ \forall N\ \exists n > N$: $\Pr\limits_{x\in U_{n+1}}(A'(f(x), 1^{n+1}) \in f^{-1}f(x)) \geq \dfrac{1}{p'(n)}$,
то есть f - не сильная односторонняя. Противоречие.

□

Теорема. *Пусть f - сильная односторонняя, регулярная по длине, g - полиномиально вычислимая, инъекция, тогда $g(f)$ - сильная односторонняя.*



Доказательство. От противного: пусть g_f^{-1} - не сильная односторонняя,
то есть $\exists A\ \exists p\ \forall N\ \exists n > N$: $\Pr\limits_{x\in U_n}(A(g_f(x), 1^n) \in g_f^{-1}g_f(x)) \geq \dfrac{1}{p(n)}$.
Построим алгоритм A' следующим образом:

1) A' получает на вход строку $y = f(x) : |y| = n + l_g(n)$ и n единичек.
2) A' применением функции g получает $z = g(y)$.
3) A' применяет алгоритм A к z, получает $x' = A(z, 1^n)$

$\Pr\limits_{x\in U_n}(A(g_f(x), 1^n) \in g_f^{-1}g_f(x)) = \Pr\limits_{x\in U_n}(A(g(f(x)), 1^n) \in g_f^{-1}g(f(x))) =$
Последняя вероятность фактически есть вероятность успеха алгоритма A' на выдачу результата из множества $g_f^{-1}g(f(U_n))$, но $g_f^{-1}g(f(x)) = f^{-1}f(x)$, так как g - инъекция.
$= \Pr\limits_{x\in U_n}(A'(f(x), 1^n) \in f^{-1}f(x))$, а значит

$\exists A'\ \exists p' = p\ \forall N\ \exists n > N$: $\Pr\limits_{x\in U_n}(A'(f(x), 1^n) \in f^{-1}f(x)) \geq \dfrac{1}{p'(n)}$,
то есть f - не сильная односторонняя. Противоречие.

□

Замечание. *Пусть выполнены условия второй теоремы о композиции. Обозначим за $\mathbb{A}_f$ множество алгоритмов, пытающихся обратить f, а за $\mathbb{A}_{g(f)}$ - пытающихся обратить $g(f)$. Тогда существует отображение $b : \mathbb{A}_{g(f)} \to \mathbb{A}_f$, определенное на всем $\mathbb{A}_{g(f)}$, такое что если $b(A_{g(f)}) = A_f$, то A_f обращает каждый из входов с той же вероятностью, что и $A_{g(f)}$.*

Доказательство. Замечание очевидно так как если возьмем алгоритм, пытающийся обратить $g(f)$, то согласно теореме 2 о композиции нужно просто предварительно применить g к результату f, и тогда вероятность того, что A_f обращает $f(x)$ есть вероятность алгоритма $A_{g(f)}$ обратить $g(f(x))$. □

Кодирование Машин Тьюринга со второй лентой

Давайте определимся со способом кодирования МТ, причем условимся, что у МТ есть вторая лента, начальное содержимое которой также зашито в коде МТ. Пусть N - длина кода МТ. Тогда первыми $\lceil \log_2 N \rceil$ символами закодируем длину кода второй ленты, которая располагается в конце кода МТ. Осталось закодировать сами правила МТ. Мы считаем без потери общности, что алфавитом всех рассматриваемых нами МТ является $\{0, 1\}$, соответственно символ, обозреваемый головкой МТ, мы можем закодировать с помощью одного бита. Аналогично с помощью одного бита кодируется направление движения головки. Однако какого количества битов достаточно для кодирования всех состояний МТ?

Лемма. *Пусть N - длина кода МТ (без второй ленты), n - количество правил, Q - число состояний, а $q = \lceil \log_2 Q \rceil$ - соответственно количество битов, достаточное для кодирования состояния. Тогда для кодирования правила МТ достаточно $2\lceil \log_2 2n \rceil + 3$ битов, и $n \geq \lfloor \frac{N}{2\lceil \log_2 2n \rceil + 3} \rfloor$.*

Доказательство. Мы хотим получить оценку на минимальное число правил МТ при данной длине ее кода N. Правил же тем меньше, чем больше количество символов q, требуемое на кодирование состояния МТ. В каждом правиле не может быть более двух состояний. Тогда, предположив, что в каждом правиле есть два различных состояния, которые не встречаются в остальных правилах, мы получим минимальную оценку на количество правил, так как получим максимальную оценку на количество битов, необходимое для кодирования состояния. Из сказанного выше мы получаем, что $Q = 2n$, тогда $q = \lceil log_2 2n \rceil$. Количество бит, необходимых для кодирования правила есть $2q + 3$, так как нужно закодировать для левой и для правой части правила состояния, обозреваемые символы, а также для правой части правила - направление движения головки. Тогда общее количество правил есть длина кода МТ, разделенная на количество бит, необходимых для кодирования одного правила: $n = \lfloor \frac{N}{2q+3} \rfloor \geq \lfloor \frac{N}{2\lceil \log_2 2n \rceil + 3} \rfloor \geq \lfloor \frac{N}{2\lceil \log_2 N \rceil + 3} \rfloor$. Последнее неравенство верно, так как $N > 2n$: правило нельзя закодировать

с помощью двух битов. Осталось заметить, что между двумя частями правила и между самими правилами никакие разделители не нужны, так как по длине записи N МТ мы можем получить длину кода состояния ($\log_2 N$ будет достаточно в любом случае, следовательно эту величину и примем за длину кода состояния), а значит можем последовательно выбирать правила из кода МТ и декодировать их. □

Итого мы получаем для МТ с лентой, что для кодирования состояния достаточно $\lceil log_2(N - \lceil \log_2 N \rceil - L) \rceil$ битов, где L - длина второй ленты.

Получение новых односторонних функций в предположении их существования

Обозначим код МТ, реализующей функцию f, за $Code(f)$.

Лемма. *Пусть односторонние функции существуют, то есть* $\exists N \exists x : |x| = N \wedge x = Code(f) \wedge f$ *- односторонняя, и пусть* g *- функция, умножающая вход на число, записанное на второй ленте, и* $|Code(g)| = \alpha$*, тогда* $\forall n : n > N + \log_2 N + \alpha$
$\Pr\limits_{x \in \{0,1\}^n}(x \in \{x_i : x_i = Code(f_i) \wedge f_i$ *- односторонняя* $\}) \geq 2^{-(N+\alpha+\log n)}$

Доказательство. Будем рассматривать строку длины n как код МТ со второй лентой. Пусть в первых $\log_2 n$ битах будет закодирована длина ленты, равная $n - \log_2 n - N - \alpha$, далее будет находиться код функции f, функции g, а в конце - лента, на которой будет закодирована константа. Тогда для любой константы, код которой умещается в длину ленты, весь код будет соотвествовать односторонней функции. Это так, так как по второй теореме о композиции $g(f)$ - сильная односторонняя, если f - сильная односторонняя, а g - инъекция. Но функция $g = c \cdot x$, где c - константа со второй ленты, x - результат работы функции f, инъекция, причем $\forall c_1, c_2 \forall x \in \mathbb{N} \quad g_{c_1}(x) \neq g_{c_2}(x)$, следовательно $\forall c_1, c_2 \forall x \in \mathbb{N} \quad g_{c_1}(f(x)) \neq g_{c_2}(f(x))$. Итак, мы зафиксировали первые $\log_2 n - N - \alpha$ бит, оставив остальные биты (а именно - биты, кодирующие число на ленте) произвольными, а фиксация такого количества битов и дает нам требуемую нижнюю оценку вероятности получения односторонней функции в предположении о ее существовании. □

Лемма. $\exists p \, \exists M : \forall n > M \Pr\limits_{x \in \{0,1\}^n}(x = Code(f) \wedge f$ *- односторонняя* $) > \dfrac{1}{p(n)}$.

Доказательство. Пусть $M = 2^k N$, тогда $\Pr\limits_{x \in \{0,1\}^M} > 2^{-(N+\alpha+k+\log_2 N)}$. Осталось проверить, что $2^{N+\alpha+k+\log_2 N} = p(M)$. Так как $N + \alpha$ - константа, то нужно проверить, что $c \cdot 2^k = p(M)$, но $c \cdot 2^k = c \cdot \dfrac{2^k \cdot N}{N} = c \cdot \dfrac{M}{N} = c' \cdot M = p(M)$. Однако, чтобы сама константа c' была полиномиальной относительно M, нужно, чтобы $\dfrac{2^N}{M^\beta} \leq 1$, или, что то же самое $2^k N \geq 2^{N/\beta}$. Для

этого достаточно равенства $k = \frac{N}{\beta}$. Заметим, что второе условие нужно исключительно на практике. То есть чтобы на практике была полиномиальность константы относительно длины входа, мы должны выбрать β и фиксировать его. Теоретически же мы можем говорить, что из условия существования односторонних функций нам известно N, соответственно мы можем взять $\beta = N$, и тогда $k = 1$. □

Как мы знаем, пока что доказано, что полные односторонние функции работают как односторонние лишь в том случае, если на ту часть их входа, которая отвечает за моделируемую машину, подать код односторонней функции. Но до недавнего времени предположение о существовании односторонней функции давало нам лишь экспоненциально малую вероятность получить одностороннюю функцию (экспоненциально малую от длины кода этой функции). А так как мы ничего не знаем о длине минимальной односторонней функции, то полные односторонние функции на практике были не интересны. В следующей теореме мы докажем, что из полной односторонней функции можно сделать одностороннюю не только с помощью передачи известного кода односторонней функции на вход полной односторонней.

Теорема. *Пусть односторонние функции существуют. Тогда любая полная односторонняя функция - односторонняя, если вход моделируемой MT использовать не только как вход, но и как сам код моделируемой MT, то есть можно считать, что $x = Code(f)$, и если U - полная односторонняя, то $U(x, x)$ - слабая односторонняя.*

Доказательство. Пусть $U(x, x)$ - не слабая односторонняя, то есть $\forall p\ \exists A'\ \forall N\ \exists n > N : \Pr\limits_{x \in U_n} (A'(U(x, x), 1^n) \in U^{-1}U(x, x)) > 1 - \frac{1}{p(n)}$ Из этого следует, что для любых $p_1(n)$ и $p_2(n)$ $\forall N\ \exists n > N$: на доле входов большей чем $1 - \frac{1}{p_1(n)}$ функция $U(x, x)$ обратима с вероятностью большей чем $1 - \frac{1}{p_2(n)}$. Если это не так, то $\exists p_1(n), p_2(n)\ \forall N\ \exists n > N : \Pr\limits_{x \in U_n} (A'(U(x, x), 1^n) \in U^{-1}U(x, x)) \leq 1 - \frac{1}{p_\alpha(n)}$, где α - индекс того полинома, для которого неравенство не выполнилось. Значение другого полинома принимаем равным бесконечности, а значит значение одного из множителей равно 1. Принимаем p_α за p, и получаем противоречие с предположением о том, что $U(x, x)$ - не слабая односторонняя.

Итак, не существует полиномиальной доли входов функции U, на которой эта функция была бы необратима в полиномиальной доле случаев. Однако доля входов, которые кодируют одностороннюю функцию, начиная с некоторой длины входа полиномиальна, а значит существует полином p^* : $\Pr\limits_{x \in U_n} (x = Code(f) \wedge f$ - сильная односторонняя$\wedge \Pr(A'(U(x, x) =$

$f(x), 1^n) \in U^{-1}U(x,x)) > 1 - \frac{1}{p_2(n)}) > \frac{1}{p^*(n)}$. Заметим, что разные x кодируют разные односторонние функции, но полиномиальную оценку доли мы получали для функций, полученных композицией с биекцией из минимальной сильной односторонней функции, а тогда можно воспользоваться замечанием к теореме 2 о композиции с односторонней, говорящее о том, что алгоритму, обращающему конкретный вход $g(f(x))$ с некоторой вероятностью, можно сопоставить алгоритм, обращающий $f(x)$ с той же вероятностью. Осталось лишь заметить, что рассматриваемому алгоритму A', обращающему односторонние функции на одном из их входов, по замечанию можно сопоставить алгоритм $A(f(x), 1^n, g)$, обращающий f на каждом из таких входов, то есть на полиномиальной доле $\frac{1}{p^*(n)}$, но тогда f - не сильная односторонняя. Противоречие. □

Алгоритм RSA и практическое применение полных односторонних функций

Итак, мы получили конкретную слабую одностороннюю функцию в предположении существования односторонних функций - универсальную МТ. Однако неизвестно, начиная с какой длины входа она действительно хорошо кодирует входы, а значит на практике она сама по себе не применима. Однако мы можем воспользоваться теоремами о композиции и получить одностороннюю функцию, которая будет приемлемо работать на входах малой длины. Построим композицию универсальной МТ и RSA.

Алгоритм RSA основан на сложности задачи разложения числа на простые множители. То есть взлом RSA сводится к полиномиальному решению задачи разложения на множители, однако открытой проблемой остается возможность обратного сведения.

Алгоритм RSA состоит из следующих шагов:

1) выбираются два простых числа p и q.

2) вычисляется $n = p * q$.

3) выбирается $e < n$: $\gcd(e, (p-1)(q-1)) = 1$.

4) вычисляется решение (d, y) в целых числах уравнения $e * d + (p-1)(q-1) * y = 1$.

5) (e, n) - публичный ключ.

6) d - приватный ключ.

7) Кодируемая строка разбивается на блоки длины $k = [\log_2 n]$. Эти блоки могут быть интерпретированы как числа $m_i \in (0, 2^k - 1)$.

8) Каждый такой блок шифруется числом $c_i = (m_i)^e \mod n$.

9) С помощью e и c_i найти m_i можно только полным перебором. Однако $((c_i)^d \mod n) = ((m_i)^{e*d} \mod n) = m_i$

Свойство. *Функция RSA - инъекция.*

Доказательство. От противного: пусть $\exists m_j, m_k : m_j \neq m_k, c_j(m_j) = c_k(m_k)$. Тогда $m_j = ((c_j)^d \mod n) = ((c_k)^d \mod n) = m_k$. Противоречие. □

Свойство. *Функция RSA действует из строк длины k в строки длины $k+1$.*

Доказательство. $c_i = ((m_i)^e \mod n) \leq n-1$

$k = [\log_2 n] < \log_2 n$, так как n - произведение двух простых чисел и случай $n = 4$ не интересен.

$2^k < n < 2^{k+1}$, и тогда $2^k - 1 < n - 1$, но c_i может быть равно $n-1$, а значит, что для кодирования c_i нужен $k+1$ символ. □

Лемма. *Пусть U - полная односторонняя функция, увеличивающая длину максимум в $c(n)$ раз. Обозначим за D_n множество возможных значений функции $U(x,x)$, где $x \in U_n$, то есть $D_n = \bigcup\limits_{k \leq c \cdot n} U_k$.*

Также пусть $\forall x \in U_n \quad \Pr(U(x,x) = y, y \in D_n) = \dfrac{1}{|D_n|}$.

Тогда $\Pr(\forall x, y \in U_n \quad U(x,x) \neq U(y,y)) > (\dfrac{2^{(c-1)n+1} - 1}{2^{(c-1)n+1}})^{2^n}$.

Доказательство. Нам нужно посчитать отношение количества удовлетворяющих нас возможных вариантов значений $U(x,x)$ к количеству всех возможных вариантов. Мы можем перейти к подсчету количества вариантов, так как предполагаем равномерное распределение на множестве значений. Заметим также, что количество всех предполагаемых возможных вариантов значений функции $U(x,x)$ есть сумма вариантов строк длины меньшей или равной $c \cdot n$, а значит равно $2^{c \cdot n+1}$.

$$\Pr(\forall x, y \in U_n \quad U(x,x) \neq U(y,y)) = \frac{2^{c \cdot n+1} \cdot (2^{c \cdot n+1} - 1) \cdot \quad \ldots \quad \cdot (2^{c \cdot n+1} - 2^n)}{2^{(c \cdot n+1) \cdot 2^n}} \geq$$

$$\frac{(2^{c \cdot n+1} - 2^n)^{2^n}}{2^{(c \cdot n+1) \cdot 2^n}} = (\frac{2^{(c-1)n+1} - 1}{2^{(c-1)n+1}})^{2^n}$$ □

Автор выражает благодарность Сергею Николенко, посоветовавшему попытаться доказать вторую теорему о композиции.

Список литературы

[1] Goldreich O. Foundations of Cryptography. Basic Tools. Cambridge University Press, 2001.

[2] Grigoriev D., Hirsch E.A., Pervyshev K.A. Complete Public-Key Cryptosystem: Tech. Rep. 006-046: Electronic Colloquium on Computational Complexity, 2006.

[3] Gurevich Y., Shelah S. Expected computation time for Hamiltonian path problem, SIAM J. Comput., 16 (1987), pp. 486-502.GHP06Gol01