*Науковий часопис НПУ імені М. П. Драгоманова. Серія 1. Фізико-математичні науки*
Київ: НПУ імені М. П. Драгоманова. — 2012, № 13 (2).— С. 77–83.

# Пример построения возможно односторонней функции на основе контексно-свободной и регулярных грамматик

С. В. Кондакова,

Национальный педагогический университет имени М. П. Драгоманова

С. В. Орлов,

Одесский национальный университет имени И. И. Мечникова

Аннотация. В статье рассматривается алгоритм построения возможно односторонней функции для построения системы шифрования с открытым ключом. Делается упор на практическую реализацию данного алгоритма. Производится подбор удобных и быстрых автоматов и грамматик.

**Ключові слова:** алгоритм, односторонняя функция, грамматика, автомат.

# Perhaps one-way function constructed via context-free and regular automata

S. Kondakova,

National Pedagogical Dragomanov University

S. Orlov,

Odessa Mechnikov National University

Abstract. This article deals with perhaps one-way function algorithms construction for building the ciphering system with the open key. The accent is being made on the practical realization of this algorithm. The article also deals with the choice of convenient and fast automata and grammars.

**AMS Subject Classifications (2010): 68Q45, 03D05.**

**Key words:** one-way function, automata, grammar, algorithm.

Будем строить, возможно, одностороннюю функцию для реализации системы шифрования с открытым ключом. Рассмотрим криптосистему, с открытым ключом шифрования использующую метод шифрования с помощью раскрашивания. Исходный текст, представленный в виде 26 латинских букв, раскрашивается краской с 26 цветами. Буквы шифруются как раскрашенные слова. Для различных появлений букв выбираются различные слова из сгенерированных множеств.

*E-mail:* s_kondakova@ukr.net, serorlov@ukr.net

Для шифрования и дешифрования будем выбирать контекстно-свободные и регулярные грамматики. В приведенной, в качестве ссылок, литературе, изучается данная проблема, например в [**1**]. Однако подобные исследования носят скорее теоретический характер. В данной работе детализируется алгоритм реализации криптосистемы. Например, специальный подбор грамматик и автоматов делает возможной эффективную практическую реализацию данной системы шифрования.

Пусть $DKA_0 = \{V_s, V_T, D_0, S\}$— конечный детерминированный автомат:

$V_s$ — множество состояний,

$V_T$ — входной алфавит,

$D_0$ — функции переключения и выхода,

$S$ — начальное состояние.

Будем рассматривать автоматы в табличной нотации. Вместо множества допускающих состояний будем использовать функции выхода двух типов {Допустить, Отвергнуть}. В автомате $DKA_0$ оставим часть клеток пустыми, не добавляя вообще функции выхода. Во всех автоматах $DKA_i$ вообще не будем использовать выход по «$\dashv$» концевому символу входного потока — оставляем соответствующий столбец пустым. Допустим, что в автомате $DKA_0$ заполнены функциями переключения половина клеток. Сформируем функции выхода $D_i$ так, чтобы каждой из них выделялось приблизительно одинаковое количество клеток автомата для выхода {Допустить}. Все оставшиеся пустыми клетки заполняются выходом = {Отвергнуть}. При этом:

1) множества клеток с функциями выхода «Допустить» не должны пересекаться для $D_i$ , $i = \overline{1, 26}$.

2a) Дополнительной гарантией для однозначного дешифрования может служить существенно более широкий алфавит $V_T$ для DKA, чем алфавит шифруемого текста. Например, можно использовать двухбайтовый Unicode. Тогда функции «допустить» для $D_i$ будут располагаться даже в разных столбцах автомата.

2b) Предположим, что функции «допустить» $D_i$ для некоторых i будут располагаться в одинаковых столбцах автомата. Тогда, при легальном дешифровании, возможна неоднозначность при выборе очередного $DKA_i$. А значит мы, вероятнее всего, на следующих дешифрованных символах, получим ошибку автомата. Это отсечет такие неоднозначности. Вероятность получения даже двух легальных и корректных версий расшифрованного текста очень мала. А вероятность получения второго осмысленного текста при легальном дешифровании исчезающе мала. На практике, в худшем случае, мы получим

несколько вариантов дешифрованного текста, включая корректный вариант. Если это когда либо произойдет, например раз в 100 лет, то повторная передача и расшифровка документа решит все проблемы.

3) Функции переключения состояний оставить одинаковыми для всех автоматов.

Зауваження 1. *Очевидно, что устойчивость ко взлому системы основанной именно на подобном варианте (2b) выбора $DKA_i$ будет существенно выше.*

Получим $DKA_i$, $i = \overline{1,26}$ множество детерминированных конечных автоматов. Приведем их. Для этого объединим эквивалентные и удалим недостижимые состояния. Итерационные алгоритмы приведения достаточно быстры. Эти 26 автоматов определяют регулярные языки $\mathrm{L}(DKA_i)$ для которых:

$\bigcap_{i=1}^{26} L(DKA_i) = \{\oslash\}$.

Это очевидно по построению. Цепочка входных символов однозначно определяет позицию в которую автомат попадет под ее воздействием. Обратное утверждение не верно. В клетку автомата может привести множество цепочек входных символов. Допустимое слово языка $DKA_i$ переключает автомат в «Допускающую» клетку. Поэтому:

a) Каждая из «Допускающих» клеток $DKA_i$ определяет множество слов языка $\mathrm{L}(DKA_i)$ — в общем случае счетное.

b) Множество слов, допускаемое другой клеткой детерминированного конечного автомата, не может включать слово допущенное функцией выхода другой клетки автомата.

Получим набор языков $\mathrm{L}(DKA_i)$ , $i = \overline{1,26}$ каждый из которых определяет непересекающиеся множества слов. Эти множества используем для кодирования i-той буквы латинского алфавита. Так как языки $\mathrm{L}(DKA_i)$ теоретически могут иметь счетную длину, для каждой конкретной подстановки слова вместо буквы мы можем подобрать множество вариантов при шифровании. Конечно, присутствует проблема удлинения шифрованного текста.

Зауваження 2. *Для данного алгоритма проблема удлинения шифрованного текста , как минимум, на порядок менее значима, чем для черно-белой раскраски битов 0,1.*

В автомате отсутствует «Допускающий» выход для символа конца входного потока «$\dashv$». Это позволяет, при дешифровании, не использовать никаких ограничивающих знаков для отделения слов, шифрующих каждый символ. Рассмотрим пример шифрования -дешифрования. Входной текст это большие латинские буквы.

Возьмем слово «ABBA». Шифровать каждую большую букву будем словами из маленьких латинских букв от «a» до «h». Для шифрования 2 букв достаточно привести только два автомата. Для «A» — $DKA_1$ и для «B» — $DKA_2$. Свободных клеток в автомате для размещения функций выхода чуть больше 30. Поэтому допускающих функций для каждой из букв будет не более двух. Найдем несколько допустимых слов для каждого автомата:

$$\mathrm{L}(DKA_1) = \{\mathrm{daf},\ \mathrm{hb},\ \mathrm{f},\ldots\}\ ,$$

$$\mathrm{L}(DKA_2) = \{\mathrm{c},\ \mathrm{hdg},\ \mathrm{dcbc},\ldots\}\ .$$

| $DKA_1$ **для входной буквы «A»** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| «A» | a | b | c | d | e | f | g | h | «⊣» |
| S | «Отв» | S | «Отв» | S1 | S2 | «Доп» | «Отв» | S1 | «Отв» |
| S1 | S | «Доп» | S2 | S1 | «Отв» | S6 | «Отв» | «Отв» | «Отв» |
| S2 | «Отв» | S | «Отв» | S4 | «Отв» | «Отв» | S3 | «Отв» | «Отв» |
| S3 | «Отв» | S2 | «Отв» | «Отв» | S5 | «Отв» | «Отв» | S6 | «Отв» |
| S4 | «Отв» | S2 | «Отв» | S6 | S5 | «Отв» | S8 | «Отв» | «Отв» |
| S5 | S2 | «Отв» | S7 | «Отв» | «Отв» | S6 | «Отв» | «Отв» | «Отв» |
| S6 | «Отв» | «Отв» | S8 | «Отв» | S2 | «Отв» | S1 | S7 | «Отв» |
| S7 | S2 | «Отв» | S | «Отв» | «Отв» | S5 | «Отв» | S2 | «Отв» |
| S8 | S2 | «Отв» | S5 | S3 | «Отв» | S5 | «Отв» | S7 | «Отв» |
| $DKA_2$ **для входной буквы «B»** | | | | | | | | | |
| «A» | a | b | c | d | e | f | g | h | «⊣» |
| S | «Отв» | S | «Доп» | S1 | S2 | «Отв» | «Отв» | S1 | «Отв» |
| S1 | S | «Отв» | S2 | S1 | «Отв» | S6 | «Доп» | «Отв» | «Отв» |
| S2 | «Отв» | S | «Отв» | S4 | «Отв» | «Отв» | S3 | «Отв» | «Отв» |
| S3 | «Отв» | S2 | «Отв» | «Отв» | S5 | «Отв» | «Отв» | S6 | «Отв» |
| S4 | «Отв» | S2 | «Отв» | S6 | S5 | «Отв» | S8 | «Отв» | «Отв» |
| S5 | S2 | «Отв» | S7 | «Отв» | «Отв» | S6 | «Отв» | «Отв» | «Отв» |
| S6 | «Отв» | «Отв» | S8 | «Отв» | S2 | «Отв» | S1 | S7 | «Отв» |
| S7 | S2 | «Отв» | S | «Отв» | «Отв» | S5 | «Отв» | S2 | «Отв» |
| S8 | S2 | «Отв» | S5 | S3 | «Отв» | S5 | «Отв» | S7 | «Отв» |

*Рис 1. Автоматы для шифрования/дешифрования — $DKA_i$ ,i=1,2.*

Можно заметить, что все слова для шифрования «A» заканчиваются на «b» или «f» шифрования «B» заканчиваются на «c», «g». Очевидно, что для $DKA_i$, выбранных с учетом замечания 2b, на эти же символы могут и заканчиваться слова, выбранные для шифрования других букв входного текста.

После шифрования «ABBA» можем получить, например, такой вариант зашифрованного текста: «dafchdbhb». Легальная дешифровка предполагает наличие у получателя автоматов $DKA_i$. Подаем на вход системы строчку «dafchdghb». Просматриваем ее с права на лево. Последний символ определяет какой из автоматов $DKA_i$ применяется.

У нас последняя буква «b», поэтому применяется последним $DKA_1$. Однозначно определяем последнее слово — оно равно «hb». А последняя расшифрованная буква — «A». Теперь последняя буква «g», поэтому применяется последним $DKA_2$. Однозначно определяем последнее слово — оно равно «hdg». А следующая расшифрованная буква — «B». Затем последняя буква «c», поэтому применяется последним $DKA_2$. Однозначно определяем последнее слово — оно равно «c». А следующая расшифрованная буква — «B». Затем последняя буква «f», поэтому применяется последним $DKA_1$. Однозначно определяем последнее слово — оно равно «daf». А следующая расшифрованная буква «A».

В системе шифрования с открытым ключом набор языков $\mathrm{L}(DKA_i)$, $i = \overline{1,26}$ является *секретным* ключом. Он будет использоваться для дешифровки или электронной подписи документа. На практике удобнее использовать не языки, а именно грамматики $DKA_i$, $i = \overline{1,26}$.

Для получения *открытого* ключа системы шифрования будем рассматривать приведенные контекстно-свободные грамматики $G_i$, $i = \overline{1,26}$. В них отсутствуют правила с бесплодными и недостижимыми нетерминальными символами. Эти грамматики используют то же множество терминальных символов, терминальный алфавит$V_T$, что и $\mathrm{L}(DKA_i)$. Очевидно, что для грамматик $G_i$ легко построить автоматы с магазинной памятью $AMP_i$, $i = \overline{1,26}$ . Эти грамматики и автоматы являются контекстно-свободными КС они входят в класс языков 2 по классификации Хомского:

$$L[AMP_i] = LKC_i \subseteq KC, i = \overline{1,26}.$$

Рассмотрим пересечение следующих языков:

$$LKC_i \cap L[DKA_i] = L[G_i], i = \overline{1,26}.$$

Так как $DKA_i$ — конечный детерминированный автомат, то он допускает регулярный язык — класс № 3. Поэтому из [**2**] [теорема 7.27] следует, что эти пересечения $\mathrm{L}[G_i]$ являются тоже контекстно-свободными языками. Полученные грамматики

$G_i = \{V_H, V_T, P_i, S\}$, $i = \overline{1,26}$ можно использовать как ключ для шифрования, где

$V_H$— множество не терминальных символов,

$V_T$ — входной алфавит,

$P_i$ — правила грамматики,

$S$ — аксиома грамматики.

Процесс получения результирующего АМР при помощи синхронизации параллельно работающего конечного автомата и автомата с магазинной памятью известен, [**2**]. Для практической реализации алгоритма шифрования удобно использовать пары синхронно работающих автоматов:

$$AMP_i \oplus DKA_i, i = \overline{1,26}.$$

Известно, что $DKA_i$ быстрый и удобный для реализации инструмент. Предлагаемый в работе $AMP_i$ ,на основе $G_i$ грамматики, легко построить. Можно оставить одинаковым количество состояний у $AMP_i$ и $DKA_i$. Более того можно оставить неизменными и функции переключения в эти автоматах. Тогда при помощи операций с магазинной памятью возможно реализовать отсеивающий слова механизм, характерный для контекстно-свободных грамматик.

Например, только на основе алгоритма балансировки скобок, можно рассмотреть целый ряд отсеивающих КС механизмов (будем считать гласные-открывающими, а согласные закрывающими скобками и т.п.). Известно, что регулярные языки являются подмножеством контекстно-свободных языков. Однако подобное усечение регулярного языка уже не будет, в общем случае, регулярным языком. Известно, что построение этих автоматов и их работа N-полная задача.

Автоматы $DKA_i$ используются в качестве секретного ключа. Легальный пользователь решает задачу принадлежности слова регулярному языку L$[DKA_i]$. Эта известная задача имеет сложность сравнимую с полиномиальным алгоритмом. Метод построения языка гарантирует, что языки L$[DKA_i]$, $i = \overline{1,26}$ не пересекаются. Поэтому дешифровка будет однозначно определенной.

Попытка взлома предполагает решение задачи принадлежности слова контекстно-свободному языку L$[G_i]$, что является «трудной» NP-полной задачей для контекстно-свободного языка [**3**]. То есть не существует алгоритм полиномиальной мощности, решающий данную задачу.

## Список литературы

[1] *Богаченко Н. Ф.,Файзуллин Р.Т.*, Автоматы, грамматики, алгоритмы, — Омск:Наследие, — 2006. — 104с.

[2] *John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman*, Introduction to automata theory, languages, and computation, — Addison-Wesley, — 2001. — 521с.

[3] *Гери М., Джонсон Д.*, Вычислительные машины и трудно разрешимые задачи, — М.:Мир,— 1982. — 419с.