Новый синтаксис ЛЯПАСа

1 Операнды

Операндами в рЛЯПАСе являются константы, переменные, комплексы и элементы комплексов. Они используются для представления неотрицательных целых чисел, булевых векторов, символов Unicode и последовательностей из них. Имеются натуральные, единичные и символьные константы. Натуральные константы записываются как десятичные, шестнадцатеричные, восьмеричные и двоичные числа. Единичная константа – это булев вектор с единственной компонентой 1. Она обозначается In если номер компоненты 1 равен n. Символьная константа – это последовательность символов Unicode. В ней символы $'$ и \ записываются как пары \$'$ и \\ соответственно. Переменными являются малые латинские буквы и буква Z, используемая для специальных целей. Комплекс есть линейно упорядоченное множество элементов, которые в символьном комплексе суть символы, а в логическом комплексе – булевы векторы длины 32, называемые также словами. Далее все эти понятия вводятся формально:
десятичная цифра ϱ ::= 0|1|2|3|4|5|6|7|8|9, десятичная константа ∂ ::= $\varrho\{\varrho\}$;
шестнадцатеричная цифра η ::= ϱ|A|B|C|D|E|F, шестнадцатеричная константа $\hbar$ ::= $\eta\{\eta\}$h;
восьмеричная цифра ω ::= 0|1|2|3|4|5|6|7, восьмеричная константа ϖ ::= $\omega\{\omega\}$o;
двоичная цифра ε ::= 0|1, двоичная константа β ::= $\varepsilon\{\varepsilon\}$b ;
натуральная константа ι ::= $\partial|\hbar|\varpi|\beta$;
символ σ ::= любой Unicode-символ, символьная константа $\Sigma ::='\sigma\{\sigma\}'$;
переменная υ ::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|Z;
единичная константа I ::= I∂|Iυ; константа ::= $\iota|\Sigma$|I;
символьный комплекс F ::= F∂, логический комплекс L ::= L∂, комплекс κ ::= F|L;
индекс J ::= $.\partial|\upsilon|(\upsilon+\partial)|(\upsilon-\partial)$; элемент комплекса ::= κJ,
значение натуральной константы ::= неотрицательное целое|булев вектор;

значение символьной константы ::= значение натуральной константы|символ;
значение переменной ::= неотрицательное целое|булев вектор;
значение элемента логического комплекса ::= неотрицательное целое|булев вектор;
значение элемента символьного комплекса ::= значение элемента логического комплек-са|символ.

В отличие от других языков программирования, типы значений в рЛЯПАСе не фиксированы. Тип значения (целое или вектор) для константы, переменной и элемента комплекса определяется типом операции (арифметической или логической соответственно), которая применяется к этому операнду. Компоненты в булевом векторе нумеруются, начиная с 0 в направлении справа налево; неотрицательные целые ограничиваются числом $2^{32}-1$, и длина булева вектора — числом 32.

2 Элементарные операции

2.1 Передача значения

Пусть, как обычно, τ есть собственная (внутренняя) переменная ЛЯПАСа, α — произвольные переменная или элемент комплекса, γ — переменная, элемент комплекса или константа, χ — значение на выходе генератора псевдослучайных чисел в компьютере (PRNG), ζ — начальное состояние PRNG, θ — значение на выходе таймера в компьютере. Тогда:
Oυ – присвоение наименьшего значения (нулей): $\upsilon ::= 00...0$;
$^{-}\upsilon$ – присвоение наибольшего значения (единиц): $\upsilon ::= 11...1$;
$\Rightarrow \alpha$ – присвоение τ: $\alpha ::= \tau$;
γ – присвоение γ: $\tau ::= \gamma$;
$=(\upsilon_1\upsilon_2)$ – обмен значениями: переменные υ_1 и υ_2 обмениваются своими значениями;
$=(\kappa\upsilon_1\upsilon_2)$ или $=(\kappa\upsilon\partial)$ и $=(\kappa.\partial\upsilon)$ – обмен значениями элементов комплекса $\kappa\upsilon_1$ и $\kappa\upsilon_2$ или $\kappa\upsilon$ и $\kappa.\partial$ соответственно;
X означает $\tau ::= \chi$; $\Rightarrow$X означает $\zeta ::= \tau$; T означает $\tau ::= \theta$.

2.2 Логические и арифметические операции

! – вычисление номера правой единицы: $\tau ::= !\tau$;
$\neg$ – отрицание $\tau ::= \neg\tau$;
% – вычисление веса булева вектора: $\tau ::= \%\tau$;
$\vee\gamma$ – дизъюнкция: $\tau ::= \tau \vee \gamma$;

$\&\gamma$ – конъюнкция: $\tau ::= \tau\&\gamma$;
$\oplus\gamma$ – сложение по модулю 2: $\tau ::= \tau \oplus \gamma$;
$< \gamma$ – левый сдвиг: $\tau ::= \tau < \gamma$;
$> \gamma$ – правый сдвиг $\tau ::= \tau > \gamma$;
$+\gamma$ – сложение по модулю 2^{32}: $\tau ::= \tau + \gamma$;
$-\gamma$ – вычитание по модулю 2^{32}: $\tau ::= \tau - \gamma$;
$*\gamma$ – умножение по модулю 2^{32}: $\tau ::= \tau * \gamma$, Z ::= переполнение;
$: \gamma$ – деление двойного числа: $\tau ::=$(Z$\tau : \gamma$), Z ::= остаток;
$/\gamma$ – частное целочисленного деления: $\tau ::= \tau/\gamma$, Z ::= остаток;
$;\gamma$ – остаток целочисленного деления: $\tau ::= (\tau;\gamma)$, Z ::= частное;
$\triangle\gamma$ – увеличение на 1: $\tau ::= \gamma ::= \gamma + 1$;
$\bigtriangledown\gamma$ – уменьшение на 1: $\tau ::= \gamma ::= \gamma - 1$.

3 Операции перехода

$\rightarrow \partial$ – безусловный переход к параграфу ∂;
$\circ\!\!\rightarrow$ – переход к параграфу ∂ по условию $\tau = 0$;
$\mapsto \partial$ – переход к параграфу ∂ по условию $\tau \neq 0$;
$\uparrow (\gamma_1 \diamond \gamma_2)\partial$ – переход к параграфу ∂ по отношению $\diamond \in \{=, \neq, <, >, \leq, \geq\}$;
$\twoheadrightarrow \partial$ – уход к параграфу ∂ с возвратом,
$\Updownarrow$ – возврат к точке, следующей за точкой ухода $\twoheadrightarrow \partial$;
$\uparrow (\gamma)\partial$ – переход к параграфу ∂ по времени γ;
$\uparrow$X$\partial\alpha_1\alpha_2$ – перечисление единиц: если $\alpha_1 = 0$, то $\rightarrow \partial$, в противном случае правая 1 исключается из α_1, ее номер $\Rightarrow \alpha_2$ и следующая операция;
{assembly program} – ассемблерная вставка: выполняется программа на языке Ассемблера, указанная между { и }.

4 Операции над комплексами

Пусть ξ is ∂ or υ; тогда:
Qξ – мощность комплекса, имеющего номер ξ;
Sξ – емкость комплекса номер ξ;
$@ + \kappa(\xi)$ – образование комплекса κ емкости ξ и мощности 0;
$@ - \kappa$ – уничтожение комплекса κ;
$@\%\kappa$ – сокращение емкости комплекса до его мощности;
Oκ – очистка комплекса (без изменения его мощности): $\kappa ::= 00...0$;
$@'\sigma_1\sigma_2...\sigma'_m >$F – символы $\sigma_1, \sigma_2, ..., \sigma_m$ добавляются к символьному комплексу F;

@ $> \kappa\xi$ – вставка элемента: значение τ вставляется в комплекс κ перед ξ-м элементом (в отсутствие ξ — после последнего элемента);
@ $< \kappa\xi$ – удаление элемента: ξ-й элемент (в отсутствие ξ — последний элемент) комплекса κ перемещается в τ, мощность комплекса уменьшается на 1;
@#$\kappa 1\kappa 2(\xi 1, \xi 2, \xi 3)$ – $\xi 1$ элементов комплекса $\kappa 1$, начиная с $\xi 2$-го элемента, копируются в $\kappa 2$, начиная с $\xi 3$-го элемента, мощности $\kappa 1, \kappa 2$ не изменяются. В отсутствие $\xi 3$ копирование производится в конец $\kappa 2$, в отсутствие $\xi 3$ и $\xi 2$ копируются первые $\xi 1$ элементов $\kappa 1$ и, наконец, в отсутствие $\xi 3$, $\xi 2$ и $\xi 1$ — все элементы комплекса $\kappa 1$.

5 Операции ввода-вывода

/F>C – вывод символьного комплекса F на консоль;
/$'\varsigma'$ >C – вывод символьной константы ς на консоль;
/F<C – ввод символьной константы с клавиатуры: вводимые символы добавляются к символьному комплексу F, мощность комплекса увеличивается.

6 Расширение рЛЯПАСа

6.1 Длинная арифметика

Натуральные константы в расширении ЛЯПАС-Т являются целыми из множества $\{0, 1, ..., 2^n - 1\}$, где n кратно 32 и зависит от фактической реализации ЛЯПАСа-Т. В настоящее время значение $n = 2^{14}$ вполне приемлемо для криптографических приложений.

Обозначая число 2^{32} символом δ, любую натуральную константу a можно выразить в виде:

$$a = a_0 + a_1\delta + a_2\delta^2 + ... + a_{k-1}\delta^{k-1} \qquad (1)$$

для некоторых $k > 0$ и $a_i \in \Delta = \{0, 1, ..., 2^{32} - 1\}$, $i = 0, 1, ..., k - 1$. В стандартном двоичном представлении элементы множества Δ являются словами — булевыми векторами длины 32. Поэтому в ЛЯПАСе-Т последовательность $a_0, a_1, ..., a_{k-1}$ представляется логическим комплексом Lj мощности k с a_i в качестве i-го элемента в Lj.

Все операции, определенные в рЛЯПАСе над переменными, в ЛЯПАСе-Т могут применяться к логическим комплексам. В случае арифметической

операции последовательность элементов комплекса рассматривается как натуральная константа a заданная формулой (1). Различные операнды для одной и той же арифметической операции могут иметь различные длины и типы (один из них – переменная, другой – комплекс). В случае логической операции значение комплекса рассматривается как булев вектор, являющийся конкатенацией элементов комплекса.

6.2 Множественность собственной переменной

Таким образом, в отличие от ЛЯПАСа, в ЛЯПАСе-Т есть два типа операндов для элементарных операций: переменные длины в одно слово и логические комплексы различных длин — от одного до $n/32$ слов. Соответственно этому, в ЛЯПАСе-Т имеются и два типа собственной переменной — простая и комплексная. Первая — из ЛЯПАСа, имеет длину одного слова. Она может принимать значения любой переменной языка. В любой реализации ЛЯПАСа-Т, программной или аппаратной, она представляется в регистре процессора. Собственные переменные 2-го типа имеют длины логических комплексов и могут принимать значения последних. В аппаратной реализации ЛЯПАСа-Т все они могут представляться в одном и том же регистре максимальной возможной длины — n разрядов. В программной реализации ЛЯПАСа-Т роль комплексной собственной переменной на время выполнения цепочки операций, начинающейся с обращения к логическому комплексу, возлагается непосредственно на этот комплекс.

В дальнейшем изложении, там, где это не вызывает двусмысленности, собственная переменная любого типа, будь то простая или комплексная, обозначается (как в ЛЯПАСе) буквой τ и называется соответственно простой или комплексной τ.

6.3 Дополнительные операции

В дополнение к операциям в рЛЯПАСе расширение ЛЯПАС-Т содержит некоторые новые логические операции, используемые в записи криптографических алгоритмов, включая следующие, где $\lambda := \upsilon|\mathrm{L}|\kappa\upsilon|\kappa\partial|\iota$:

1) $\Updownarrow\mathrm{L}$ – перестановка: компоненты булева вектора τ переставляются в соответствии с их порядковыми номерами, указанными в элементах логического комплекса L;

2) $\_(\xi 1, \xi 2)$ – проекция: выбирается часть булева вектора τ с компонентами, имеющими номера в интервале $(\xi 1, \xi 2)$;

3) $\Uparrow \xi 1\lambda(\xi 2, \xi 3)$ – вставка: часть булева вектора λ с компонентами,

имеющими номера в интервале $(\xi 2, \xi 3)$ вставляется в τ перед $\xi 1$-й компонентой (в отсутствие $\xi 3$ вставляется правая часть булева вектора λ длины $\xi 2$; в отсутствие $(\xi 2, \xi 3)$ вставляется весь булев вектор λ);

4) $\Downarrow (\xi 1, \xi 2)$ – редукция: часть булева вектора τ с компонентами, имеющими номера в интервале $(\xi 1, \xi 2)$ вычеркивается из вектора;

5) $\mid \lambda$ – конкатенация: булев вектор λ присоединяется к τ ;

6) $\ll \xi 1(\xi 2, \xi 3)$ or $\gg \xi 1(\xi 2, \xi 3)$ – левый или правый циклические сдвиги: часть булева вектора τ с компонентами, имеющими номера в интервале $(\xi 2, \xi 3)$ циклически сдвигается на $\xi 1$ бит влево или вправо соответственно (в отсутствие $\xi 3$ сдвигается правая часть булева вектора λ длины $\xi 2$; в отсутствие $(\xi 2, \xi 3)$ сдвигается весь булев вектор λ).

Что касается арифметических операций по модулю N (для некоторого натурального N), таких, как сложение и вычитание mod N, умножение mod N, возведение в степень mod N и другие, широко используемые в криптографии, фактически нет никакой реальной возможности для включения их в список элементарных операций ЛЯПАСа-Т из-за существования великого множества различных алгоритмов их выполнения, имеющих разные эффективности во многих различных случаях. Вместо этого решено эти алгоритмы реализовывать по мере надобности и возможности на ЛЯПАСе-Т и (или) на языке Ассемблера и включать их в библиотеку ЛЯПАСа-Т для использования в программах на ЛЯПАСе-Т в качестве подпрограмм.

7 Компилятор ЛЯПАСа-Т

Далее для краткости любая программа на ЛЯПАСе-Т и ее подпрограммы называются L-программой и L-подпрограммами соответственно.

Для выполнения L-программы компьютером она должна быть подана в качестве параметра компилятору, который преобразует ее в загрузочный модуль (исполняемый код) для ОС Linux. Компилятор запускается по команде последней

<user>: $ lc <prog>.l

где <prog>.l — это имя файла с L-программой, являющейся списком L-подпрограмм. (Рекомендуется файлу с L-программой давать имя с расширением l, но это не обязательно). Первая в списке L-подпрограмма является головной. ОС Linux передает ей управление после загрузки файла в оперативную память компьютера. Порядок следования других L-подпрограмм в списке несущественный. Файл может содержать не все

необходимые L-подпрограммы. В этом случае компилятор находит недостающие в файле libl0.l, являющемся библиотекой пользователя. В ней рекомендуется хранить наиболее часто используемые L-подпрограммы.

Результатом работы компилятора является загрузочный модуль, который хранится под именем <prog> (без расширения) в той же папке, где находится и L-программа. Запуск скомпилированной программы на выполнение осуществляется по команде
<user>: $ ./<prog>

Компилятор написан на языке C++ с использованием библиотеки регулярных выражений, делающим его максимально простым и прозрачным.

7.1 Структура загрузочного модуля

Загрузочный модуль состоит из двух сегментов – сегмента программы (.text) и сегмента данных (.data). В свою очередь, первый сегмент состоит из подпрограмм, генерируемых компилятором для L-подпрограмм, вызываемых в процессе исполнения L-программы. Сегмент данных содержит: текущий адрес в памяти для размещения новых комплексов и границу памяти, отводимой ОС под комплексы; текущее состояние PRNG; единичные константы; веса всех булевых векторов в $\{0,1\}^8$ (нужны для реализации операции взвешивания %); и все символьные константы, встречающиеся в L-программе.

7.2 Организация памяти

Для каждой L-подпрограммы все ее локальные переменные, начала, мощности и емкости ее локальных комплексов размещаются в стеке, образующем фрейм в 1420 байтов. Доступ к локальным данным осуществляется по фиксированным смещениям от начала фрейма (регистр ebp). Значение регистра ebp фрейма родительской подпрограммы также сохраняется во фрейме, образуя список фреймов вызванных L-подпрограмм.

Локальные комплексы L-подпрограммы размещаются в "куче". Адрес свободного участка кучи на момент вызова подпрограммы также сохраняется во фрейме.

Создание локального комплекса сопровождается проверкой свободного места путем сравнения величин емкости создаваемого комплекса, адреса свободного участка и границы памяти, отводимой под комплексы. Если места достаточно, то адрес начала комплекса получает значение адреса свободного участка, а адрес свободного участка увеличивается на значение

емкости комплекса. Если места не достаточно, то выполняется обращение к ОС для увеличения границы доступной памяти.

Такая организация памяти защищена от атак переполнения стека и кучи, поскольку, во-первых, буферы (комплексы) убраны из стека вместе с возможностью переписать адрес возврата и, во-вторых, нет операций для освобождения "кучи"посредством ОС.