УДК 004.056.5

ДОСЛІДЖЕННЯ СТРУКТУРИ ДАНИХ ДЕКАРТОВЕ ДЕРЕВО

Науковий керівник – д.т.н, проф., Лужецький В. А.

Стах О. С.

Важливість структур даних важко переоцінити. Адже правильно організоване збереження даних в пам’яті дає змогу значно пришвидшити роботу програм. Виходячи з цих міркувань можна зробити висновок, що використання структур даних для захисту інформації є актуальною задачею.

Існує досить багато структур даних,всі їх можна поділити на 2 основні групи: лінійні (стек, масив), нелінійні(логарифмічні: куча, дерево). Перевагою лінійних структур даних є можливість виконання деяких операцій за O(1) часу (наприклад, індексація для масиву). Проте деякі операції виконуватимуться за O(n) часу (видалення елементу). Логарифмічні структури зберігають час при виконанні певних операцій над даними, хоча не кожна з них може бути виконана на логарифмічних структурах. Наприклад при реалізації бінарного дерева пошуку чи кучі не можна швидко знайти суму на підмасиві даних чи розвернути частину масиву. Також проблемою подібних структур є можливе розбалансування, при якому висота структури може стати лінійною. В такому випадку зручно використовувати Декартове дерево. Це вид бінарного дерева, вузол якого, крім посилань на ліве та праве піддерева містить два ключі – x (ключ) та y (пріорітет). По x – це являється двійковим деревом пошуку, по y – двійковою кучею. Така організація дозволяє виконувати дві основні операції – поділ дерева по ключу та злиття двох дерев – за час $O(\log_2 n)$, де n – висота дерева. Будь-які операції над деревом зводяться до даних двох. Єдина умова, що накладається на злиття дерев – ключі (x) лівого дерева повинні бути меншими за ключі правого дерева. Тому різновидом Декартового дерева є дерево по неявному ключу – аналог масиву чисел. Тобто при злитті двох дерев ми передаємо їх як параметри в такому порядку, в якому нам потрібно їх розташування у вихідному масиві. Пріорітети (y) використовуються лише для балансування.

Таким чином можна використати наступний алгоритм для захисту:

1. Розбиття повідомлення на n блоків, нумерація їх в природному порядку.
2. Задання n пріоритетів у випадковому порядку.
3. Побудова декартового дерева розмірності n по пріоритетам.
4. За допомогою генератора псевдовипадкових чисел згенерувати індекси [st; fin]
5. Виконати інверсію даних на підвідрізку [st; fin]
6. Повторювати п.4 до забезпечення необхідної стійкості.
7. Перебудувати повідомлення згідно результуючої нумерації блоків
8. Провести гамування для виключення атаки нав’язуванням тексту.

Таким чином ми добились захищеності даних.