DOI: 10.20535/kpisn.2023.1-2.263225
УДК 004.6

Бондарчук Максим Юрійович[1*], Тесленко Олександр Кирилович[1]
[1]Факультет прикладної математики, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
[*]Відповідальний автор: bondarchuk.m.y@gmail.com

ДОСЛІДЖЕННЯ ЛІНІЙНОЇ ФУНКЦІЇ ПІДСТАНОВКИ ДОВІЛЬНОЇ РОЗРЯДНОСТІ У КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕННЯХ

Проблематика. Криптографічні перетворення завжди викликали інтерес освіченої частини людства і є невід'ємною частиною сучасних комунікацій. Є різні криптографічні алгоритми для різних задач з різними вимогами. Функції підстановок є корисними для випадків, коли швидкість перетворень є важливішою за теоретичну секретність. Апаратна реалізація таких підстановок є доволі простою.
Мета дослідження. Оглянути модель, комбінаційні схеми для апаратної реалізації та алгоритм програмної реалізації функції підстановки. Дослідити алгоритми атак і підбору досліджуваної функції підстановки.
Методика реалізації. Описано алгоритми криптографічних перетворень та їх криптоаналіз для бієктивних підстановок, реалізованих за допомогою регулярних комбінаційних структур лінійної складності. Запропонована функція забезпечує швидкість перетворень даних до гігабіт на секунду. Уточнюється алгоритм формування елементів регулярних структур підстановки, об'єми відкритих та секретних даних. Розглянуто формати даних, методи їх передачі та апаратна реалізація одного з методів. Описано види атак та алгоритми підбору схем елементів регулярної структури підстановки з експериментальним обчисленням кількості необхідних операцій. Розроблено програмну реалізацію запропонованих алгоритмів для обчислення результатів.
Результати дослідження. Отримано числові результати кількості ключів, об'єму необхідної пам'яті для апаратної реалізації та кількість необхідних операцій для проведення криптоаналізу досліджуваної підстановки.
Висновки. Результати показують, що запропонована функція підстановки має достатній рівень захисту від атак трьох типів, маючи при цьому велику швидкість шифрування та розшифрування і нескладні апаратні вимоги.
Ключові слова: комбінаційні структури; криптоаналіз; функції підстановок; цифрові автомати.

Вступ

Криптографічні алгоритми використовувалися протягом багатьох століть для захисту таємниць та достовірності дипломатичного та політичного листування, військових комунікацій тощо. Більшість історичних криптографічних схем можна розглядати як схеми шифрування. Вони перетворюють повідомлення у криптограму за допомогою інверсної операції, яка залежить від секретного ключа. Дешифрування відносно просте для тих, хто володіє ключем і вважається неможливим (або дуже складним) для тих, хто не володіє ключем. Однак часто перехоплені повідомлення фактично можуть бути реконструйовані з криптограми криптоаналітиком без попереднього знання секретного ключа. Це називають криптоаналізом. Безпечна обробка інформації за допомогою криптографії стає все більш важливою.

За умови обмеженої розрядності вхідних даних програмні та апаратні реалізації прямих та обернених підстановок прості. Підстановки забезпечують бієктивне відображення і широко використовуються в алгоритмах криптографічних перетворень. Однак складність реалізації підстановок загалом пропорційна експоненті від кількості розрядів вхідних даних. Тому в алгоритмах розповсюджених криптографічних перетворень використовуються підстановки малих розрядів − здебільшого 8- або 4-розрядні [1], [2], [3].

Одним із можливих напрямів спрощення реалізації підстановок є використання структур

Пропозиція для цитування цієї статті: М. Ю. Бондарчук, О. К. Тесленко, "Дослідження лінійної функції підстановки довільної розрядності у криптографічних перетвореннях", *Наукові вісті КПІ*, № 1–4, с. XX–XX, 2023. doi: 10.20535/kpisn.2023.1-2.263225

Offer a citation for this article: M. Bondarchuk, O. Teslenko, "Use of implementations of arbitrary bitness permutations for cryptographic transformations", *KPI Science News*, no. 1–4, pp. XX–XX, 2023. doi: 10.20535/kpisn.2023.1-2.263225

© Автор(и).
Стаття поширюється на умовах ліцензії CC BY 4.0

лінійної складності від кількості розрядів вхідних даних. До таких структур належать одновимірні каскади конструктивних модулів (ОККМ). Такі структури можуть бути реалізовані як програмно, так і апаратно. Під час апаратної реалізації ОККМ є комбінаційними схемами, що розширюють можливості підвищення швидкості перетворень. Уперше структури такого класу розглядались в [4].

Під час апаратної реалізації конструктивний модуль (КМ) має дві комбінаційні схеми. Перша з них (f_o) формує значення з множини вихідних символів Y на первинних виходах КМ, друга (f_s) формує значення із множини станів S на бокових виходах КМ.

Принцип роботи полягає в тому, що структурі ОККМ, яка складається із r конструктивних модулів подається вхідне значення розрядністю $r|X|$, а на виході формується значення розрядністю $r|Y|$. За $X = Y$ ОККМ може реалізувати підстановку на множині $X_r = X \times X \times ... \times X \ (r \to \infty)$.

У [5] показано, що на найпростіших ОККМ ($|X| = |S| = 2$) можна реалізувати 48 різних підстановок будь-якої розрядності. Різні підстановки отримуються вибором булевих функцій від двох змінних f_o, f_s та початкового значення s_0, яке надходить на боковий вхід першого КМ.

У [6] було звернуто увагу на те, що функціонування апаратно реалізованого ОККМ тотожно функціонуванню відповідного цифрового автомата Мілі за будь-яких скінченних множин X, Y та S. Такий автомат має функцію переходів $\dot{f}_s(x,s): S \times X \to S$, функцію виходів $\dot{f}_o(x,s): S \times X \to Y$ і початковий стан $s_0 \in S$. Автомат реалізує відображення $X_r \to Y_r \ (r \to \infty)$. Це дозволяє використовувати властивості цифрових автоматів для визначення можливостей реалізації підстановок на множині X_r. Згідно з [7] автоматне відображення $X_r \to Y_r$ бієктивне тоді і тільки тоді, коли автомат не містить станів з втратами, які досягаються з початкового стану s_0 за r кроків. Станом з втратами називають стан s_l такий, для якого існують $x_1, x_2 \in X$, де $x_1 \neq x_2$ і $f_o(s_l, x_1) = f_o(s_l, x_2)$ [7]. Фактично це означає, що за будь-якого конкретного $s_0 \in S$. дочірня функція $f_o(s_a, X)$ є підстановкою на множині X. Це є достатньою умовою для апаратної або програмної реалізації на основі ОККМ підстановок довільної розрядності. Як показано експериментально [6], реалізація багаторозрядних підстановок на ОККМ не потребує значних витрат і різниться великою швидкістю перетворень. Теоретично швидкість формування такого відображення може складати гігабіти на секунду. Виникає запитання, яким чином використати ці переваги в швидкості для криптографічних перетворень та оцінити криптостійкість таких використань.

Постановка задачі

Дослідити використання функції підстановки довільної розрядності, запропонованої в [6], для криптографічних перетворень, дослідити захищеність від різних можливих атак на ключ.

Визначення можливих алгоритмів криптографічних перетворень на багаторозрядних підстановках

Бієктивне відображення множини вхідних послідовностей будь-якої довжини у множину вихідних послідовностей тієї ж самої довжини можна розглядати як криптографічне перетворення. Алгоритм шифрування досить простий. Мовою C# базовий алгоритм шифрування може бути поданий таким чином:

```
var outputs = new List<int>();
for (int i = 0; i < r; i++)
{
outputs
.Add(OutputMatrix[state][input[i]]);
  state = StateMatrix[state][input[i]];
}
return outputs;
```

Відкритими даними є сам алгоритм, кількість елементів n у вхідному алфавіті, кількість m в алфавіті станів – розміри *OutputMatrix* та *StateMatrix* відповідно ($m \times n$ кожна).

Ключовою (секретною) інформацією в таких криптографічних перетвореннях є функції виходів та переходів автомата. Окремого розгляду потребує значення початкового стану.

Для забезпечення максимальної швидкості реалізації бієктивних відображень доцільно кількість n елементів вхідного і вихідного алфавітів цифрового автомата вибирати із ряду степені двійки: 2^2, 2^4, 2^8, 2^{16}, 2^{32} і т. д.

За деяких конкретних m та n алгоритм легко реалізується на Асемблері. Наприклад, за $m = n = 256$, на процесорах 80x86 Intel Corporation [8] у 32-розрядному режимі реалізація базового алгоритму може виконуватись з граничною швидкістю – 4 команди на байт повідомлення.

```
MOV CL,State
@10:
;block 1
MOV BH,inputs[esi]
MOV AL,OutputMatrix[ebx+ecx]
;block2
MOV Outputs[edi],AL
MOV CL,StateMatrix[ebx+ecx]
;block3
INC EDI
INC ESI

CMP ESI,Input_End
JB @10
```

Дійсно, команди, позначені у блоках, можуть виконуватись одночасно.

За не дуже великих m та n алгоритм можна реалізувати апаратно на сучасних програмованих логічних інтегральних схемах (ПЛІС). Наприклад, за $m = n = 256$, для реалізації комбінаційних схем одного КМ знадобиться 8 булевих функцій від 8 змінних, тобто 32 таблиць пошуку (англ. look up table або LUT) ПЛІС Virtex 6 (Virtex 7) або Spartan 6 (Spartan 7) [9]. У конкретних випадках кількість LUT на один КМ може бути зменшеною під час використання спільної декомпозиції булевих функцій [4]. Секретною інформацією є біт-потік для налаштування ПЛІС.

У цій роботі розглянуто структури ОККМ, що подані на рис. 1.

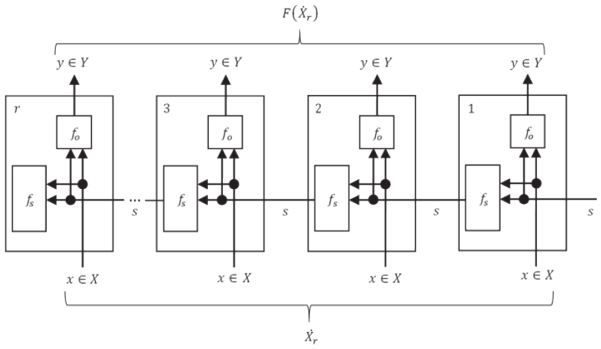

Рис. 1. Однонаправлений ОККМ [6]

У базовому криптографічному перетворенні алгоритми шифрування і розшифрування однакові. Різниця полягає лише в реалізації функцій переходів та виходів КМ. Фактично ключі шифрування та розшифрування різні. В [6] показано, як на основі функцій КМ прямого перетворення формуються функції виходів і переходів КМ оберненого перетворення. Алгоритми формування функцій оберненого КМ прості, тобто вони не потребують значних обсягів обчислень. Це не дає можливості використання базового алгоритму для криптографічних перетворень з відкритим ключем.

Алгоритм генерування таблиці виходів автомата залишається без змін, порівняно з [6]. Алгоритм генерування таблиці переходів (зв'язаний пронумерований однонаправлений граф – це граф, з будь-якої вершини якого можна потрапити до всіх його інших вершин) складається з таких кроків:

1. Згенерувати n шляхів завдовжки в одну вершину s_0 – перший символ алфавіту станів S, занести їх до масиву P. Під шляхом матимемо на увазі послідовність вершин графа, що відповідає таблиці переходів, яка генерується, наприклад $\{s_0, s_3, s_2\}$.

2. Для кожного рядка таблиці переходів (всього m, поточний – i):

   а) для кожного символу вхідного алфавіту X (всього n, поточний – j):

      і) згенерувати довільний стан S за допомогою генератора псевдовипадкових чисел;

      іі) додати до всіх шляхів масиву P, які закінчуються на i (тобто поточний рядок таблиці переходів, який ми генеруємо) згенерований стан S.

3. Якщо масив P не містить хоча б один шлях, довжина унікальних елементів якого дорівнює m, а його перша вершина дорівнює останній, то повернутись до кроку 1.

Пропонуються такі методи встановлення початкового стану:

А. По відкритому каналу, разом з шифрограмою передається вектор ініціалізації, який містить будь-які випадкові значення із множин X та S. Приймаюча сторона по функції станів та вектору ініціалізації створює значення початкового стану.

В. Початкові стани є частиною секретного ключа і вони кожного разу автоматично використовуються в криптографічних перетвореннях під час кожного сеансу зв'язку.

С. Початкові стани не є частиною секретного ключа. Під час кожного запуску додатка шифрування початковий стан зі сторони джерела даних вибирається випадковим чином. Для пересилання приймачу відповідного початкового стану використовується той чи інший криптографічний протокол на основі асиметричних криптографічних перетворень з відкритим ключем.

D. Початкові стани не є частиною секретного ключа. Ті чи інші початкові стани установлюються лише за умови організації зв'язку. Надалі як початкові стани використовуються поточні стани автоматів.

Кількісні характеристики базового алгоритму – це кількість ключів та об'єми таблиць (матриць). Кількість ключів визначається як добуток кількості різних матриць виходів, кількості різних матриць переходів та, в методі B, – кількості початкових станів.

Кількість різних матриць виходів – це кількість розміщень із $n!$ до m. Кількість різних матриць переходів – це кількість зв'язних орієнтованих графів.

Кількість зв'язаних пронумерованих графів (послідовність цілих чисел A001187 [10]) обчислюється за рекурентною формулою [11]:

$$C_m = \sum_{k=1}^{m-1} \left(\frac{m-2}{k-1} \right) \left(2^k - 1\right) C_k C_{m-k}$$
$$\left(C_0 = 1; C_1 = 1\right). \qquad (1)$$

З урахуванням кількості можливих підстановок у [6] та (1), кількість ключів для силової атаки апроксимується як

$$\frac{C_m m(n!)!}{(n!-m)!} \le O(n,m) \le \frac{m^{nm+1}(n!)!}{(n!-m)!}. \qquad (2)$$

Об'єми необхідної пам'яті для таблиць КМ показано в табл. 1.

Наближені межі кількості ключів для деяких значень параметрів конструктивних модулів показано в табл. 2.

Оскільки формула (2) вимагає довгої арифметики для обчислень граничних значень, то наведені значення нижньої оцінки дещо занижені (мінімальна кількість біт, щоб описати отримане десяткове число довгої арифметики мінус один), а верхньої оцінки – дещо завищені (мінімальна кількість біт, щоб описати отримане десяткове число довгої арифметики).

Таблиця 1. Об'єм необхідної пам'яті для деяких значень таблиць КМ

n	d, біт	m	w, біт	Q, байт	Q, МБ	Q, ГБ
4	2	4	2	8	0,000007868	0,000000008
8	3	8	3	48	0,000046134	0,000000045
16	4	16	4	257	0,000244617	0,000000239
256	8	256	8	131073	0,125000954	0,000122071
4096	12	4096	12	50331650	48	0,046875001

Таблиця 2. Наближені межі кількості ключів для деяких значень параметрів n та m

n,	m,	Нижня оцінка	Нижня оцінка, біт	Верхня оцінка	Верхня оцінка, біт
4	4	38763648	25	4,38128E+15	52
4	8	4,02E+20	68	1,95E+38	128
4	16	O(1055)	184	O(1093)	308
16	4	O(1055)	187	O(1098)	324
16	16	O(10250)	832	O(10523)	1737
16	32	O(10603)	2006	O(101185)	3935
16	64	O(101462)	4859	O(102660)	8836
32	16	O(10577)	1917	O(101199)	3982
32	32	O(101284)	4266	O(102677)	8891
32	64	O(103001)	9972	O(105936)	19717
64	32	O(102875)	9552	O(105968)	19825
64	64	O(106311)	20965	O(1013103)	43526
64	128	O(1014406)	47856	O(1028596)	94993
64	256	O(1033052)	109797	O(1062039)	206086
128	64	O(1013854)	46022	O(1028670)	95239
128	128	O(1030043)	99803	O(1062122)	206364
128	256	O(1067336)	223686	O(10133939)	444935

Закінчення табл. 2.

n,	*m*,	Нижня оцінка	Нижня оцінка, біт	Верхня оцінка	Верхня оцінка, біт
256	16	O(1013238)	43976	O(1013277)	44105
256	32	O(1018895)	62769	O(1028799)	95666
256	64	O(1032638)	108422	O(1062270)	206855
256	128	O(1065017)	215985	O(10134106)	445490
256	256	O(10139602)	463751	O(10287605)	955400

Апаратна реалізація

На відміну від програмної реалізації, коли один і той самий КМ (автомат Мілі) використовується скільки завгодно раз, в апаратній реалізації використовується регулярна апаратна структура з обмеженою кількістю КМ.

Для налаштування ПЛІС системою автоматизованого проєктування і розрахунку (САПР) створюється бітовий потік, який зазвичай записують у флеш-пам'ять. Цей бітовий потік у нашому випадку і є секретним ключем. З метою захисту інтелектуальної власності бітові потоки шифрують розширеним стандартом шифрування AES [1], а в контролерах завантаження ПЛІС (наприклад, за допомогою інтерфейсу спільної групи тестів (англ. Joint Test Action Group або JTAG) [12]) бітовий потік розшифровується.

Сучасні ПЛІС мають значну але скінченну кількість LUT. Тому у цьому разі необхідно застосовувати додаткові апаратні вузли . Нехай d – кількість КМ, а D – довжина повідомлення, яке необхідно зашифрувати. За $D > d$ одночасно можна зашифрувати лише d символів вхідного повідомлення. Таку частину будемо називати кадром, а все повідомлення розподіляється на кадри. Здебільшого останній кадр буде не заповнений. За аналогією зі стандартами блочного шифрування кадр можна доповнювати нулями. Після розшифрування з повідомлення видаляються (якщо це потрібно) останні нульові символи.

Функціональну схему для методу встановлення початкового стану B показано на рис. 2. Період тактового сигналу для регістру поточного стану має дві частини. По першій з них регістр поточного стану видає на комбінаційну схему КМ значення стану, а по другій – відбувається запис стану на виході комбінаційної схеми через мультиплексор. Мультиплексор у процесі шифрування передає в регістр поточного стану значення на виході комбінаційної схеми, а по сигналу початку шифрування – з регістру початкового стану в регістр поточного стану. Вважається, що частота даних на вході схеми менша за частоту спрацювання одного КМ комбінаційної схеми.

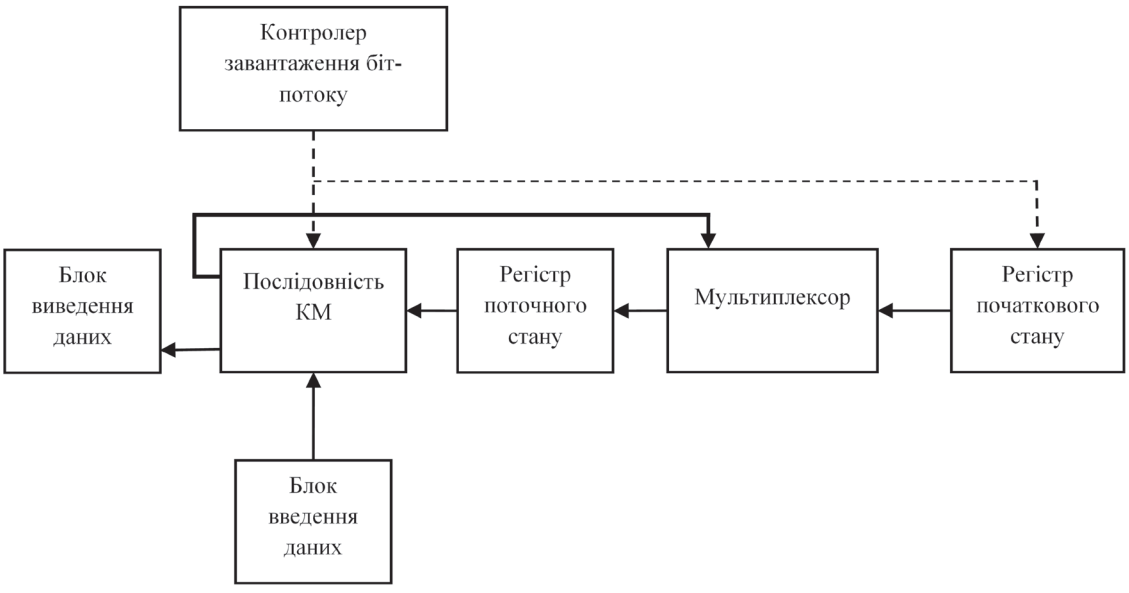

Рис. 2. Функціональна схема для методу B встановлення початкового стану

Атаки на ключ у разі відомого шифротексту

Нехай у розпорядженні криптоаналітика є криптограма $y_r \in \dot{Y}_r$.

Використаємо алгоритм, подібний на запропонований у свій час Шенноном [13] для блочних симетричних криптографічних перетворень.

Позначимо, як і раніше, $\dot{X}_r$ множину всіх послідовностей завдовжки r із елементів вхідного алфавіту, а $\dot{Y}_r$ – вихідного. Автомат Мілі, побудований згідно з [6] виконує бієктивне відображення множини $\dot{X}_r$ у множину $\dot{Y}_r$ ($\dot{X}_r \to \dot{Y}_r$).

На практиці використовують не всі послідовності множини $\dot{X}_r$. Позначимо $D_r \in \dot{X}_r$ множину допустимих послідовностей завдовжки r із елементів вхідного алфавіту. Позначимо $N_r = \frac{\dot{X}_r}{D_r}$. Оскільки відображення $\dot{X}_r \to \dot{Y}_r$ бієктивне, то існує множина E_r така, що $N_r \to E_r$.

За допомогою повного перебору всіх можливих ключів можна встановити множину недопустимих ключів K_e при заданому значенні r. Будь-яка перехоплена криптограма $y_r \in E_r$ дозволяє відбракувати всі ключі із множини K_e. За допомогою аналізу для $r' = 2,...,r$ за достатньо великого значення r теоретично можна знайти ключ, але обсяги обчислень занадто великі. Тому в розрізі цієї атаки запропоноване криптографічне перетворення є обчислювально стійким.

Атаки на основі відомого відкритого тексту та відомого відповідного шифротексту

У цьому випадку використовується алгоритм силової атаки. Оцінювання кількості ключів для силової атаки наведено в табл. 2. Згідно з таблицею, вже за $m = n = 16$ обсяги обчислень дуже значні. Тому в розрізі цієї атаки запропоноване криптографічне перетворення є обчислювально стійким, але у разі відомого відображення $x_r \to y_r$ ($x_r \in \dot{X}_r$, $y_r \in \dot{Y}_r$) завжди можна побудувати автомат для реалізації такого відображення. Питання полягає в тому, чи буде отриманий автомат еквівалентний оригінальному. Велике значення має і характер повідомлення та його довжина. Цю проблему розглянуто далі.

Атаки на основі підібраних відкритих текстів або криптограм

У запропонованій криптосистемі алгоритми шифрування та розшифрування збігаються. Різниця між ними полягає лише в заповненні таблиць виходів та переходів. Тому атаки на основі підібраних відкритих текстів не будуть різнитись від атак на основі підібраних криптограм. Можливість цих атак залежить від наповненості таблиць, тобто визначення автомата.

Проблема визначення автомата за результатами експериментів над ним активно досліджувалась і досліджується у значній кількості робіт, наприклад [14–17]. Практичною метою таких досліджень є створення мінімально можливих тестових послідовностей в умовах масового виробництва мікропроцесорних пристроїв, при цьому часто використовувався автомат-зразок, який функціонував без помилок. Важливим елементом таких досліджень є можливість встановлення для автомата одного і того самого початкового стану.

Оскільки, в нашому випадку, кожен рядок таблиці виходів автомата є унікальним, то всі алгоритми визначення автомата ґрунтуються на цій вимозі і відтворюють таблиці рядок за рядком, перевіряючи їх унікальність.

Під час оцінювання складності алгоритму визначення автомата однією операцією вважатимемо подання символу вхідного алфавіту (і, відповідно, перехід автомата в новий стан та отримання символу вихідного алфавіту на виходах), скидання автомата в початковий стан та переведення автомата в заданий стан (за наявності можливості такої операції).

Алгоритм підбору 1 (a_1)

Якщо можливе переведення автомата в будь-який заданий стан, а також скидання автомата в початковий стан, то автомат можна визначити за таким алгоритмом [18].

1. Визначення таблиці виходів. Виконується прохід за всіма можливими станами (всього m, поточний – s_i). Для кожного стану виконується прохід за всіма можливими входами (всього n, індекс поточного – j) і формується один рядок r таблиці виходів: для кожної ітерації на вхід автомата подається відповідний вхідний символ x_j (n операцій для кожного стану), вихідний символ записується в рядок r зліва направо (r_j), а автомат переводиться у стан s_i (n операцій для кожного стану). Отриманий рядок r заноситься до таблиці виходів. Таким чином таблиця виходів визначається за $m \cdot 2n = 2mn$ операцій з автоматом.

2. Визначення таблиці переходів. Виконується прохід за всіма можливими станами (всього m, поточний – s_i). Для кожного стану

виконується прохід за всіма можливими входами (всього n, індекс поточного – j) і формується один рядок r таблиці станів. Для формування кожного стану r_j виконується ще один прохід (тобто n^2 операцій для кожного стану) за всіма можливими входами (всього n, індекс поточного – k) і формується рядок таблиці виходів o_j: для кожної ітерації автомат переводиться у стан s_i (n^2 операцій для кожного стану), на вхід подається спочатку x_j (вихідний символ автомата ігнорується) (n^2 операцій для кожного стану), потім x_k (вихідний символ записується в o_{j_k}) (n^2 операцій для кожного стану). Після цього r_j визначається як індекс рядка o_j у таблиці виходів. Таким чином таблиця виходів визначається за $m \cdot 3n^2 = 3mn^2$ операцій з автоматом.

Бачимо, що для цього типу автомата його можна визначити за $a_1 = 2mn + 3mn^2(m+1)$ операцій з автоматом. Цей алгоритм відповідає методу А. Якщо в кожному рядку таблиці станів усі стани різні, то за допомогою цього алгоритму можна визначити автомат і методом В.

Алгоритм підбору 2 (a_2)

Якщо можливе скидання автомата в початковий стан, то автомат можна визначити за таким алгоритмом [19].

1. Визначення першого рядка таблиці виходів. Виконується прохід за всіма можливими вхідними символами (всього n, індекс поточного – j). Поточний вхідний символ подається на вхід автомата (n операцій), вихід автомата записується в комірку j першого рядка таблиці виходів, після чого автомат скидається в початковий стан (n операцій). Отриманий перший рядок таблиці відповідає початковому стану автомата. Таким чином перший рядок таблиці виходів визначається за $2n$ операцій з автоматом.

2. Усі інші рядки таблиці виходів, а також таблиця станів знаходяться за допомогою проходу за всіма можливими комбінаціями вхідних символів (далі – шлях, оскільки комбінація описує шлях проходу за всіма рядками таблиці станів) завдовжки m – кількість рядків таблиці станів. Кількість таких шляхів – n^m, оскільки для кожного з m елементів шляху може бути використано n різних вхідних символів. Підшляхом шляху назвемо частину шляху від його початку до будь-якого елемента шляху. Наприклад, для шляху $\{0,1,2,3\}$ підшляхами будуть $\{0\}$, $\{0,1\}$, $\{0,1,2\}$ та $\{0,1,2,3\}$. Для кожного підшляху виконати:

a) якщо цей підшлях є першим, то стан s вважати початковим (відлік з 0);

b) виконати прохід по підшляху (індекс поточного – i). Для кожного стану з підшляху отримати рядок r таблиці виходів цього стану таким чином: виконати прохід від його початку до поточного стану i (подати на вхід автомата вхідні символи з частини підшляху) (i операцій), після чого виконати прохід за всіма можливими вхідними символами (всього n, індекс поточного – j). Поточний вхідний символ подається на вхід автомата (n операцій), вихід автомата записується в комірку r_j рядка таблиці виходів, після чого автомат скидається в початковий стан (n операцій), і кожного разу виконується прохід по підшляху, але вже для наступного вхідного символу. Якщо для будь-якого підшляху вже був побудований рядок таблиці виходів, то використовується попередній результат. Інакше кажучи, для кожного підшляху виконується $n(i+2)$ операцій. Для підшляху завдовжки i виконується $n(i+2)n^i$. Оскільки виконується прохід за всіма підшляхами всіх довжин від 1 до m, причому для кожного підшляху прохід виконується один і лише один, то кількість операцій для всіх підшляхів можна обчислити як $n\sum_{i=1}^{m}(i+2)n^i$;

c) якщо отриманого на кроці a рядка таблиці виходів ще немає в таблиці виходів, то він додається до таблиці виходів;

d) визначити новий стан s^* як індекс отриманого рядка r у таблиці виходів;

e) у комірку таблиці станів з індексом рядка s та індексом стовпчика, що відповідає останньому символу підшляху, занести стан s^*;

f) замінити значення s на s^*.

Отриманий у результаті автомат може бути еквівалентним оригіналу, а може повністю збігатися. З погляду криптоаналізу це неважливо.

Видно, що для цього типу автомата його можна визначити за $a_2 = 2n + n\sum_{i=1}^{m}(i+2)n^i$ операцій з автоматом.

Алгоритм підбору 3 (a_3)

Якщо перевести автомат в будь-який заданий стан неможливо, а також неможливо скинути автомат в початковий стан, то алгоритм визначення автомата є наближеним до алгоритму повного перебору і матиме високу обчислювальну складність (NP-повна задача).

Кількість необхідних операцій згідно з алгоритмами ключів для деяких значень параметрів КМ показано в табл. 3.

Оскільки формула визначення a_2, аналогічно до формули (2), вимагає довгої арифметики для обчислень граничних значень, то наведено неточне значення кількості операцій, а нижню і верхню границю (так само як і вище: мінімальна кількість біт, щоб описати отримане десяткове число довгої арифметики мінус один і мінімальна кількість біт, щоб описати отримане десяткове число довгої арифметики відповідно).

Таблиця 3. Кількість необхідних операцій для підбору автомата згідно з алгоритмами

n	m	a_1, операцій	Нижня оцінка a_2, операцій	Верхня оцінка a_2, операцій
4	4	232	O(1039)	O(1044)
4	8	456	O(1069)	O(1074)
4	16	904	O(10126)	O(10130)
16	4	3232	O(1073)	O(1077)
16	16	12832	O(10239)	O(10243)
16	32	25632	O(10455)	O(10459)
16	64	51232	O(10883)	O(10887)
32	16	50240	O(10295)	O(10299)
32	32	100416	O(10564)	O(10569)
32	64	200768	O(101099)	O(101103)
64	32	397440	O(10674)	O(10678)
64	64	794752	O(101315)	O(101319)
64	128	1589376	O(102594)	O(102598)
64	256	3178624	O(105148)	O(105153)
128	64	3162368	O(101531)	O(101535)
128	128	6324480	O(103022)	O(103027)
128	256	12648704	O(106002)	O(106007)
256	16	3154432	O(10465)	O(10469)
256	32	6308352	O(10893)	O(10897)
256	64	12616192	O(101747)	O(101751)
256	128	25231872	O(103451)	O(103455)
256	256	50463232	O(106856)	O(106860)

Видно, що вже для малих значень n та m, а особливо після $m = n = 16$ кількість операцій набуває астрономічного значення, обчислення часу виконання яких є складною задачею і вимагає значних обчислювальних потужностей.

Висновки

Запропонована функція підстановки може застосовуватись для криптографічних перетворень і може бути реалізована програмним, програмно-апаратним або апаратним способом. Найбільш доцільно її застосовувати під час швидкого зашифрованого передавання великих об'ємів даних.

З результатів обчислень складності проведення атак трьох видів бачимо, що досліджувана функція є захищеною.

Найбільшу швидкість криптографічних перетворень має алгоритм з використанням операції виняткової диз'юнкції (англ. Exclusive OR або XOR) та використанням ключа, створеного криптостійким генератором випадкових чисел [20]. Однак ключ має бути іншим для кожного сеансу, а довжина ключа в бітах повинна відповідати довжині початкового повідомлення. У запропонованих алгоритмах ключ є великим, але обмеженого постійного розміру.

Нині спостерігається стрімкий розвиток безпілотних апаратів (БПА), які використовують смуги передачі різної частоти. Цивільні пристрої зазвичай використовують радіозв'язок, а військові – супутниковий зв'язок. Шифрування відбувається як за допомогою симетричних алгоритмів шифрування, так і асиметричних. Для таких пристроїв запропонований алгоритм може стати у нагоді через швидкість перетворення і суттєву криптографічну стійкість.

Подальші дослідження стосуються визначення автомата під час методів встановлення початкового стану C та D. Згідно з [20] є випадки, коли визначення автомата є NP-повною задачею.

На перший погляд диференційний та лінійний криптоаналіз навряд чи мають ефективне застосування для розглянутих алгоритмів, але це також може бути метою аналізу надалі.

Список літератури

[1] National Institute of Standards and Technology (2001) "Advanced Encryption Standard (AES)". *Federal Information Processing Standards.* DOI 10.6028/NIST.FIPS.197.

[2] National Institute of Standards and Technology (1999) "Data Encryption Standard (DES)": *Federal Information Processing Standards.*

[3] DSTU 7624:2014. National Standard of Ukraine. Information technologies. Cryptographic Data Security. Symmetric block transformation algorithm. Ministry of Economic Development and Trade of Ukraine, 2015 (in Ukrainian).

[4] Корнейчук В.И., Тарасенко В.П., Тесленко А.К. Исследование сложности реализации функций на одномерном каскаде модулей // Автоматика и вычислительная техника, 1977, № 5, с. 5–11.

[5] Tarasenko, V.P., Teslenko, O.K. & Yanovska O.Y. (2010). "Vlastyvosti povnykh pidstanovok, yaki realizuyut'sya nayprostishym odnonapravlenym rehulyarnym OKKM". [Properties of complete permutations implemented by the simplest unidirectional regular OCSU]. *Radioelektronni i komp'yuterni systemy*. No. 6, pp. 123–128 (in Ukrainian).

[6] Teslenko, O.K & Bondarchuk, M.Y. (2020) "Implementation of arbitrary bitness permutations in one of the classes of linear structures". *Herald of Advanced Information Technology*, Vol. 3, No. 1, 2020, pp. 406–417. DOI 10.15276/hait01.2020.7.

[7] Gill A. Introduction to the theory of finite-state machines. New York, Toronto, Ontario, London, McGraw-Hill Book Co., Inc., 1962. 207 p.

[8] Intel Processors and Chipsets by Platform Code Name [Electronic resource]. – Available at: https://www.intel.com/content/www/us/en/design/products-and-solutions/processors-and-chipsets/platform-codenames.html. – Active link: 15.09.2021.

[9] Summary of Virtex-6 FPGA Features. Virtex-6 Family Overview. XILINX DS150 (v2.5). [Electronic resource]. – Available at: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf. – Active link: 20.08.2015.

[10] Sequence A001187 – Number of connected labeled graphs with n nodes. The On-Line Encyclopedia of Integer Sequences® (OEIS®). [Electronic resource]. – Available at: http://oeis.org/A001187. – Active link: 29.03.2021.

[11] Nijenhuis, A. & Wilf, H. (1979). "The enumeration of connected graphs and linked diagrams", *Journal of Combinatorial Theory*. Series A, Vol. 27, Issue 3, pp.356–359. DOI 10.1016/0097-3165(79)90023-2.

[12] IEEE Standard Test Access Port and Boundary Scan Architecture 1149.1-2001. [Electronic resource]. – Available at: https://ieeexplore.ieee.org/servlet/opac?punumber=7481. – Active link: 27.03.2008.

[13] Шеннон К. "Работы по теории информации и кибернетике", М., ИЛ, 1963, с. 333–369

[14] Bhattacharyya, A. (1989). "Checking experiments on sequential machines". *New York: J. Wiley and Sons*, p. 155.

[15] Кудрявцев В. Б., Грунский И. С., Козловский В. А. Восстановление автоматов по фрагментам поведения. Дискрет. матем. 21:2 (2009), 3–42; Discrete Math. Appl., 19:2 (2009), 113–154.

[16] Kudryavtsev, V. B., Grunskii, I. S. & Kozlovskii, V. A. (2010). "Analysis and Synthesis of Abstract Automata". *Journal of Mathematical Sciences*. Vol. 169, Issue 4, pp. 481–532. DOI: 10.1007/s10958-010-0058-z.

[17] Petrenko, A. F., Yevtushenko, N. & Dsouli, R. (1994). "Grey box finite state machine base testing strategies". *Department of Computer Science and Operations Research*. University of Montreal. Publication No. 991, p. 22.

[18] Initial Resettable Machine Definition Algorithm. [Electronic resource]. – Available at: https://github.com/MaxBondarchuk/PermutationCryptanalysis/blob/master/PermutationCryptanalysis/InitialResettableTest.cs. – Active link: 07.06.2021.

[19] Non-initial Resettable Machine Definition Algorithm. [Electronic resource]. – Available at: https://github.com/MaxBondarchuk/PermutationCryptanalysis/blob/master/PermutationCryptanalysis/NonInitialResettableTest.cs. – Active link: 07.06.2021.

[20] Delfs, H. & Knebl, H. (2007). "Computationally Perfect Pseudorandom Bit Generators. Introduction to Cryptography". *Information Security and Cryptography*. Springer. Vol. 41, No. 4, pp. 42–44. ISBN 978-3-540-49243-6. DOI 10.1145/1907450.1907523.

Maksym Bondarchuk, Oleksandr Teslenko

USE OF IMPLEMENTATIONS OF ARBITRARY BITNESS PERMUTATIONS FOR CRYPTOGRAPHIC TRANSFORMATIONS

Background. Cryptographic transformations have always aroused the interest of the educated part of humanity and are an integral part of modern communications. A lot of different cryptographic algorithms exist for different tasks and requirements. Permutation functions are useful for cases where transformation speed is more critical than theoretical secrecy. Hardware implementation of such substitutions is quite simple.

Objective. Investigate the model and combinational circuits for hardware implementation. Investigate algorithms for permutation functions software implementation. Investigate attack algorithms and cracking of permutation functions for cryptanalysis.

Methods. The paper reviews algorithms of cryptographic transformations and their cryptanalysis for bijective permutations implemented by means of regular combinational structures of linear complexity. The proposed algorithms provide the rate of processing up to gigabits per second. The paper clarifies the algorithm of formation of elements of regular structures of permutations, specifies volumes of public and private data, reviews data formats, methods of their transfer and hardware implementation of one of the methods. The paper reviews attack types and permutation regular structure schemes cracking algorithms with experimental calculation of necessary operations quantity. The software implementation of the proposed algorithms for results calculation was developed.

Results. Numerical results of the number of keys, the amount of memory required for hardware implementation and the number of required operations for cryptanalysis were obtained.

Conclusions. The results show that the proposed algorithms for cryptographic transformations have a sufficient level of protection with a high-speed encryption and decryption.

Keywords: combinational circuits; cryptanalysis; permutation functions; finite state machines.

Рекомендована Радою
факультету прикладної математики
КПІ ім. Ігоря Сікорського

Надійшла до редакції
25 травня 2023 року

Прийнята до публікації
11 грудня 2023 року