УДК 621.391:519.2 **DOI:10.30837/rt.2024.2.217.08**

О. О. КУЗНЕЦОВ, д-р техн. наук, М. О. ПОЛУЯНЕНКО, канд. техн. наук,
Д. І. ПРОКОПОВИЧ-ТКАЧЕНКО, канд. техн. наук, Є. В. КОТУХ, канд. техн. наук,
В. О. ЛЮБЧАК, канд. фіз.-мат. наук

МОДИФІКОВАНІ ГЕНЕТИЧНІ АЛГОРИТМИ ДЛЯ ГЕНЕРАЦІЇ S-BOXES З ВИСОКОЮ НЕЛІНІЙНІСТЮ

Вступ

S-boxes (Substitution boxes, або блоки заміни) – це фундаментальні елементи криптографічних алгоритмів, які відіграють ключову роль у забезпеченні безпеки та стійкості систем шифрування [1 – 3]. Ці таблиці використовуються для заміни блоків даних на інші блоки, тим самим ускладнюючи аналіз і розшифрування закодованої інформації. Зазвичай, S-boxes приймають на вхід кілька бітів і видають на вихід кілька бітів, перетворюючи вхідні дані у вихідні через складні нелінійні функції [4 – 6].

Генерація S-boxes є критично важливим завданням у криптографії, оскільки безпека більшості сучасних криптографічних алгоритмів, таких як AES, DES, Blowfish, Twofish та багатьох інших, значною мірою залежить від стійкості та надійності їх S-boxes [3, 7, 8]. Ці блоки заміни забезпечують необхідну нелінійність та дифузію, що захищає дані від різних типів атак, включаючи лінійний та диференціальний криптоаналіз.

Створення надійних S-boxes є складним завданням, оскільки атакуючі постійно вдосконалюють свої методи криптоаналізу, намагаючись знайти слабкі місця у криптографічних системах. Тому S-boxes повинні мати специфічні властивості, які забезпечують їх стійкість до різних атак [9 – 11]:

1. **Нелінійність**: S-box повинен бути високонелінійним, щоб зміна одного біта у вхідному блоці призводила до зміни більше одного біта у вихідному блоці. Це ускладнює застосування лінійного криптоаналізу.
2. **Рівномірність**: Вихідні значення S-box повинні бути розподілені рівномірно, що забезпечує рівноймовірність кожного можливого виходу та захист від диференціального криптоаналізу.
3. **Стійкість до атак**: S-box повинен бути стійким до різних атак, таких як лінійний та диференціальний криптоаналіз, а також до методів зворотного аналізу.
4. **Інваріантність**: S-box повинен бути інваріантним до перестановки вхідних або вихідних бітів, що підвищує його стійкість до атак, заснованих на перестановці бітів.
5. **Різноманітність**: S-box повинен бути унікальним і відрізнятися від інших S-boxes, щоб запобігти атакам з використанням попередньо обчислених таблиць.

Забезпечення всіх цих властивостей при генерації S-boxes є важливою задачею для забезпечення надійності та безпеки криптографічних алгоритмів. Існує кілька методів генерації S-boxes, включаючи випадкову генерацію, методи лінійних перетворень, комбінаційну оптимізацію та генетичні алгоритми. Кожен з цих методів має свої переваги та недоліки, і вибір методу залежить від конкретних вимог до системи [12 – 14].

1. **Випадкова генерація**: Найпростіший метод, який, однак, може не забезпечити достатньої стійкості та безпеки.
2. **Метод лінійних перетворень**: Забезпечує хорошу стійкість до лінійного криптоаналізу, але може бути вразливим до інших атак.
3. **Комбінаційна оптимізація**: Дає можливість створити S-boxes з високою стійкістю до різних атак, але є обчислювально складним.
4. **Генетичний алгоритм**: Забезпечує ефективну генерацію S-boxes зі значною стійкістю до атак, при цьому є обчислювально ефективним.

Застосування генетичних алгоритмів для генерації S-boxes є перспективним напрямком досліджень, оскільки вони дозволяють досягти високого рівня стійкості при помірній обчислювальній складності. У цій статті розглядається генетичний алгоритм для генерації S-boxes. Ми досліджуємо ефективність базової версії алгоритму, демонструємо високу складність генерації нелінійних підстановок, а також пропонуємо новий алгоритм селекції, що підвищує ефективність генерації. Результати експериментів показують, що запропонований метод значно підвищує ефективність і надійність S-boxes.

2. Генетичні алгоритми для генерації S-box

2.1. Загальні відомості про генетичні алгоритми

Генетичні алгоритми (ГА) є потужним методом оптимізації, який базується на принципах природного відбору та генетики. Вперше запропоновані Джоном Холландом у 1970-х роках [15, 16], ці алгоритми моделюють процес еволюції популяцій організмів з метою пошуку оптимальних або близько оптимальних розв'язків складних задач.

Основні етапи генетичних алгоритмів включають [16, 17]:

1. **Ініціалізація популяції**: Генетичний алгоритм починається зі створення початкової популяції індивідів (можливих розв'язків). Кожен індивід кодується у вигляді хромосоми, яка може бути бінарною стрічкою, вектором чисел чи іншою структурою даних.
2. **Оцінка пристосованості (фітнес-функція)**: Кожен індивід оцінюється за допомогою фітнес-функції, яка вимірює якість або пристосованість індивіда до вирішення задачі.
3. **Відбір**: На основі фітнес-значень відбираються індивіди, які братимуть участь у створенні нової популяції. Частіше відбираються індивіди з вищою пристосованістю.
4. **Кросовер (схрещування)**: Відбрані індивіди (батьки) схрещуються між собою для утворення нових індивідів (нащадків). Кросовер дозволяє комбінувати генетичну інформацію батьків і створювати нові розв'язки.
5. **Мутація**: Застосовується випадкова зміна генів у хромосомах нащадків для підтримки генетичної різноманітності популяції і запобігання передчасній збіжності до локальних оптимумів.
6. **Заміна**: Старе покоління індивідів замінюється новим поколінням, і процес повторюється, поки не буде досягнуто критерію зупинки, наприклад, максимальна кількість поколінь або задовільний рівень фітнесу.

2.2. Застосування генетичних алгоритмів для генерації S-box

Генерація S-boxes за допомогою генетичних алгоритмів є перспективним підходом завдяки їх здатності знаходити складні, нелінійні і стійкі до атак структури. Основні етапи застосування ГА для генерації S-boxes включають [11, 18, 19]:

1. **Ініціалізація популяції S-boxes**: Початкова популяція складається з випадково згенерованих S-boxes. Кожна S-box представлена у вигляді хромосоми, де кожен ген відповідає певному значенню таблиці заміни.
2. **Оцінка пристосованості S-boxes**: Для кожної S-box обчислюється фітнес-функція, яка враховує кілька критеріїв, таких як нелінійність, рівномірність розподілу, стійкість до диференціального та лінійного криптоаналізу. Ця функція може бути комбінацією різних метричних показників.
3. **Відбір найбільш пристосованих S-boxes**: Використовуються методи відбору, такі як турнірний відбір або відбір за пропорцією пристосованості, для вибору батьківських S-boxes для наступних поколінь.
4. **Кросовер і мутація**: Відібрані S-boxes піддаються операціям кросоверу і мутації для утворення нових S-boxes. Кросовер може здійснюватися шляхом обміну частин хромосом між батьківськими S-boxes, а мутація – шляхом випадкових змін окремих значень у таблиці заміни.

5. **Заміна і еволюція**: Нові покоління S-boxes замінюють старі, і процес повторюється, поки не буде досягнуто оптимальних значень фітнес-функції або максимальна кількість поколінь.

Застосування генетичних алгоритмів для генерації S-boxes дозволяє ефективно знаходити рішення, що відповідають вимогам до стійкості та безпеки криптографічних алгоритмів. Генетичні алгоритми забезпечують широкий простір пошуку та гнучкість в налаштуванні параметрів, що дозволяє оптимізувати процес генерації S-boxes для конкретних криптографічних потреб.

3. Методологія досліджень та результати

3.1. Методологія досліджень

Для дослідження ефективності генетичних алгоритмів у генерації S-boxes була обрана цільова функція WHS (Walsh–Hadamard Spectrum), запропонована Кларком у роботі [20]. Ця функція має вигляд

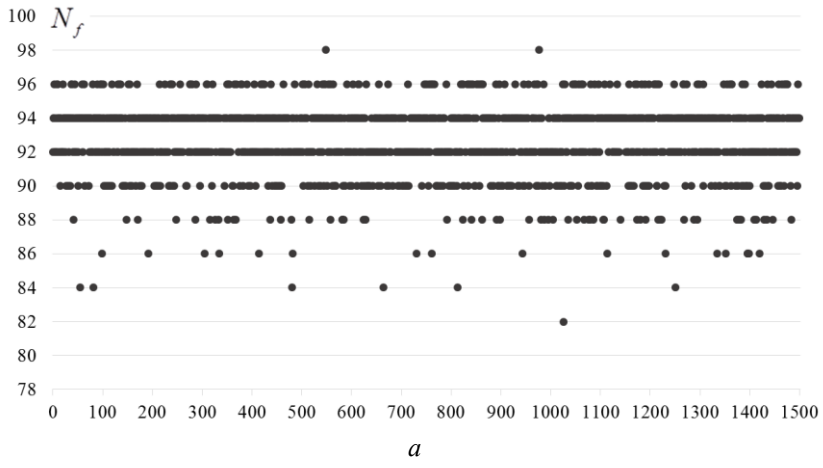
$$WHS = \sum_{b=1}^{255}\sum_{i=0}^{255} \left| |WHT[b,i]| - X \right|^R ,$$

де WHT (англ. Walsh–Hadamard transform) – спектральні коефіцієнти Уолша–Адамара; i – цикл за всіма компонентними функціями та їх лінійними комбінаціями; b – цикл за всіма лінійними функціями; X і R – параметри з дійсними значеннями.

В якості оптимальних параметрів функції *WHS* було обрано $R = 12$ та $X = 0$. При оптимальних параметрах та використанні алгоритму пошуку сходження на пагорб з ймовірністю близької до 99 % може бути сформований бієктивний S-блок з нелінійністю 104 [21, 22].

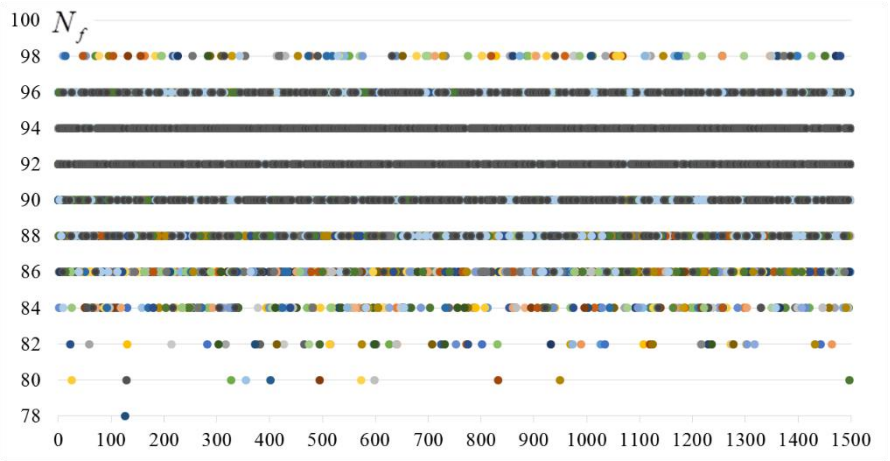
Генетичний алгоритм передбачає вибір найкращого результату серед нащадків. У початку алгоритму пошуку дуже швидко можливо знайти покращення результату при випадковому виборі мутацій стану S-блоку. Але чим кращий поточний стан S-блоку тім складніше знайти покращення та майже завжди всі зміни S-блоку будуть мати гірше значення.

Ми провели ряд запусків генетичного алгоритму з цільовою функцією *WHS* з найкращими її параметрами $R = 12$ та $X = 0$, кількість ітерації була обмежена значенням у 150 000, кожні 100 ітерацій фіксувався поточний стан: значення цільової функції та нелінійність. В якості прикладу на рис. 1, *а* наведено отримані послідовність зміни нелінійності для одного запуску алгоритму пошуку. На рис. 1, *б* наведено результат 50 таких запусків. Для кожного запуску ми використовували п’ять різних мутацій та кількість екземплярів у популяції дорівнювало п’яти. Як бачимо, жодного разу нелінійність не досягла хоча б значення у $N_f = 100$.



а

Рис. 1. Зміна нелінійності через кожні 100 ітерацій (всього 150 000 ітерацій):
а – результат одного запуску, *б* – результат за 50 запусками



б

Рис. 1. (Продовження)

На рис. 2 наведено гістограму розподілу частоті нелінійності для тих же результатів. На рис. 3 наведено частоту нелінійності для випадково сформованого S-блоку отриманих на вибірки з 10^8 формувань.

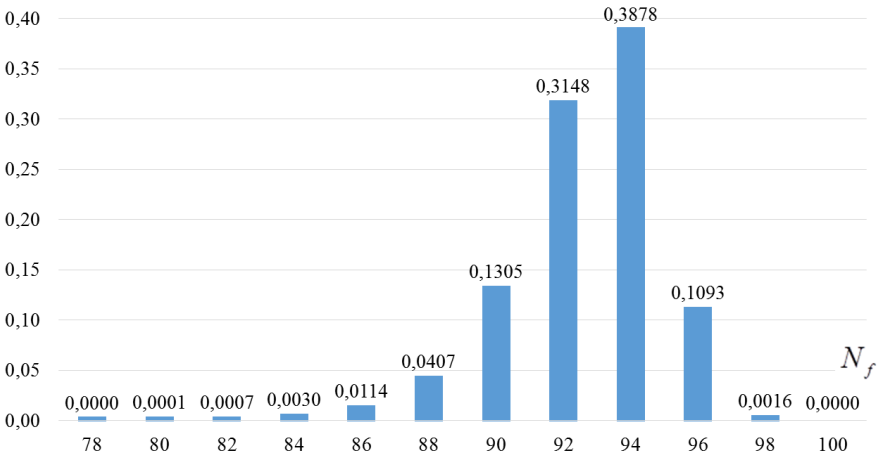


Рис. 2. Гістограма розподілу нелінійності за резльутатами 75 000 замеров впродовж 50 окремих запусків алгоритму пошуку

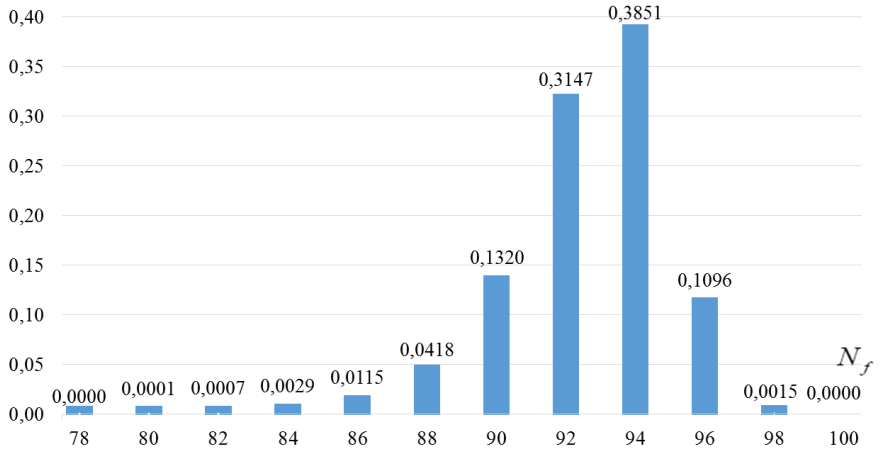


Рис. 3. Розподіл кількості сформованих випадковим чином S-блоків в залежності від їх нелінійності (N_f) при 10^8 випробуваннях

Як бачимо, розподіли майже ідентичні, що вказує на відсутність прогресу покращення пошуку генетичним алгоритмом у його «класичному» виді та обраних параметрів пошуку. Тому ми дещо змінили алгоритм пошуку, додавши зберігання до популяції не лише нащадків, а ще й поточну популяцію. Відбраковування у популяції зайвої кількості S-блоків будемо проводити за принципом відбору найкращих за значенням цільової функції. Данна модифікація алгоритму пошуку за функціональністю стала походити на методи Hill Climbing.

3.2. Алгоритм селекції

Основна ідея методу селекції полягає у розгляді деякої популяції S-блоків, яка формується випадковим чином. Проведення мутації у цієї популяції. Під мутацією будемо розуміти зміну містами двох випадково обраних (неоднакових) позицій у S-блоці; оцінку якості отриманих нових S-блоків за допомогою деякої цільової функції; ранжирування S-блоків зі старої популяції та нових S-блоків; створення нової популяції найкращих екземплярів S-блоків згідно зі створеним ранжируваним списком S-блоків (обираючи до нової популяції найкращих представників); повторення наведених дії до тих пір, поки не буде знайдено цільовий S-блок або не виконано інші критерії зупинки алгоритму.

Псевдокод алгоритму селекції наведено на рис. 4.

Вхід: S_{pop}, K_{iter}, K_{pop}
For ($t = 0$ to $t < K_{\text{iter}}$, $t = t + 1$){
    $S_{\text{pop}} = \text{basic\_selection}\left[S_{\text{pop}} \right]$;
    For ($p = 0$ to $p < K_{pop}$, $p = p + 1$){
        $S \leftarrow S_{\text{pop}}[p]$;
        For ($k = 0$ to $k < K_{\text{mut}}$, $k = k + 1$){
            $S' \leftarrow S$;
            $i \leftarrow random[0...255]$;
            $j \leftarrow random[0...255]$;
            $swap(S'[i], S'[j])$;
            $N_f, F_c \leftarrow f(S')$;
            If ($N_f \le 104$) Return S';
            $S_{\text{pop}} = S_{\text{pop}} + S'$;
        }
    }
}
Return 0.

Рис. 4. Псевдокод алгоритму селекції

На вхід алгоритм селекції отримує:

- K_{pop} – кількість екземплярів у популяції – кількість найкращий S-блоків яку зберігаємо у популяції;
- S_{pop} – випадковим чином сформована популяція бієктивних S-блоків. В нашому випадку, для початкового формування бієктивного S-блоку, ми використовували алгоритм Фішера – Йетса;
- K_{iter} – максимальна кількість ітерацій виконання алгоритму селекції. Є одним з критеріїв зупинки алгоритму, якщо він не в змозі подолати локальний мінімум;

- K_{mut} – кількість мутацій – кількість нових екземплярів (S-блоків), отриманих з кожного представника (S-блоку) з поточної популяції.

Ітеративно виконуємо крок алгоритму селекції. Виконуємо не більше $K_{\text{iter}} = 150000$ ітерацій. На початку кожної ітерації виконуємо селекцію у популяції за допомогою функції basic_selection:

basic_selection – функція, яка виконує ранжирування списку S-блоку за значенням цільової функції та значенням нелінійності. До топу заносяться S-блоки, які мають більш високу нелінійність, S-блоки, які мають однакову нелінійність сортуються за значенням цільової функції: чим менше значення – тим ближче до топу поміщується S-блок. На виході функції залишається S-блоки з топу у кількості K_{pop} штук.

З кожним окремим екземпляром популяції з S_{pop} виконуємо алгоритм мутації K_{mut} раз. В якості алгоритму мутації нами обрана заміна містами двох випадково вибраних (неоднакових) елементів у S-блоці. Одночасно обчислюєтеся її цільова функція (F_c) та перевіряється значення нелінійності (N_f). Якщо нелінійність відповідає бажаному значенню, то завершуємо алгоритм селекції та повертаємо знайдений S-блок. У іншому випадку додаємо до популяції новий створений екземпляр. Наприкінці роботи кроку ітерації маємо популяцію розміром $K_{\text{pop}} \times (K_{\text{mut}} + 1)$, яку на наступній ітерації, за допомогою функції basic_selection, зменшимо до розміру K_{pop}.

3.3. Отримані результати

Враховуючи, що обчислення значення цільової функції є сама затратна (з точки зору процесорного часу) операція, складність обчислювання всього алгоритму пошуку можна вважати пропорційною кількості викликів обчислення цільової функції. Тобто кількості S-блоків, які було сформовано та перевірено. Позначимо таку кількість як K_{Sbox}.

Для прискорення роботи алгоритму було проведено паралельне обчислення нової популяції у $N_{thread} = 8$ потоках всередині кожної ітерації.

Алгоритм селекції виконувався при різних значеннях $K_{\text{pop}} = [1,21]$ (з кроком 2) та $K_{\text{mut}} = [1,31]$ (з кроком 3). Для кожного значення виконувалось по 100 запусків алгоритму пошуку.

Усереднене значення складності пошуку наведено у таблиці 1 та візуалізовано на рис. 5. Найкращі параметри, з точки зору меншої кількості обчислень цільової функції, були при $K_{\text{pop}} = 1$. Усереднене значення K_{Sbox} коливається у діапазоні від 49 277 до 58 213. Слід зазначити, що при кожному випробуванні мале місце дуже велике коливання від середнього значення (приклад для кращого значення у $K_{Sbox} = 49277$ при $K_{\text{pop}} = 1$ та $K_{\text{mut}} = 7$, наведено на рис. 6). Середньоквадратичне відхилення для $K_{\text{pop}} = 1$ складало 22 500 а для $K_{\text{pop}} = 21$ – 44 500.

Враховуючи найкращі результати при малих значеннях K_{pop}, для K_{pop}=1 та 2, було виконано ще серію зі 100 запусків для $K_{\text{mut}} = [1,30]$ (з кроком 1). Результати наведено на рис. 7. При цьому середньостатистична кількість K_{Sbox} становить близько 53 000 для $K_{\text{pop}} = 1$.

Майже завжди алгоритм пошуку знаходив S-блок з нелінійністю $N_f = 104$. Лише у 2 % для випадку $K_{\text{pop}} = 1$ та $K_{\text{mut}} = 1$ алгоритм зупинився у локальному мінімумі, не досягнувши мети пошуку, та у 1 % – для значень $K_{\text{pop}} = 1$ та $K_{\text{mut}} = 6,8,11$.

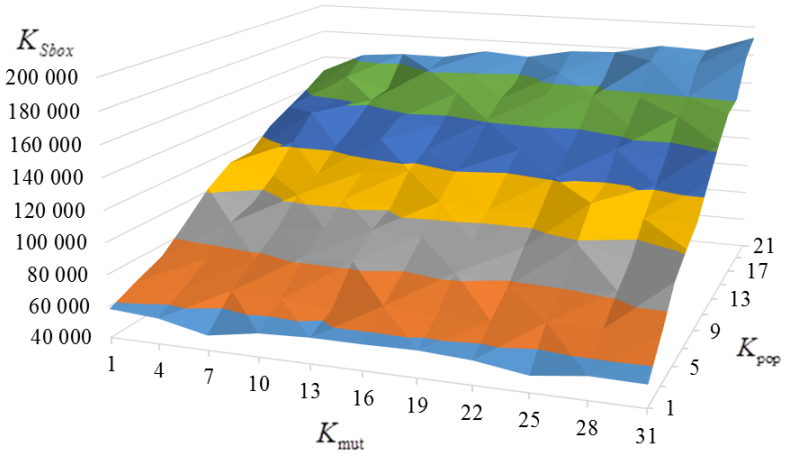


Рис. 5. Усереднене значення кількісті сформованих S-блоків (K_{Sbox}) до знаходження S-блоку з $N_f = 104$

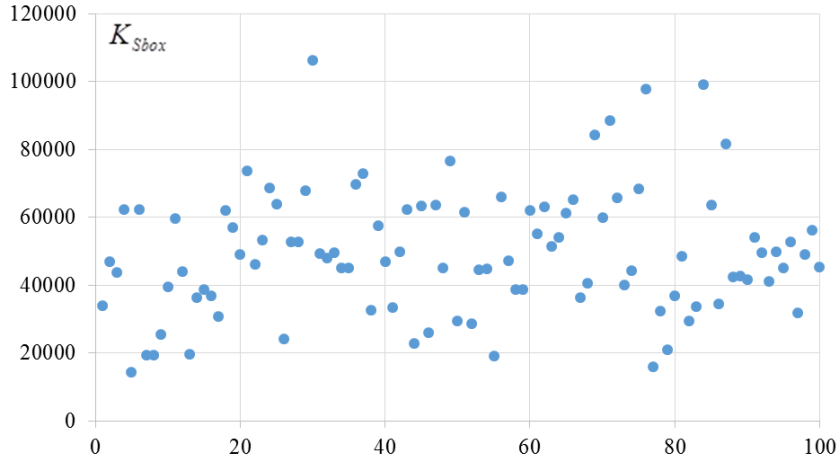


Рис. 6. Кількість S-блоків, які було сформовано та перевірено (K_{Sbox}) до знаходження $N_f = 104$ при $K_{pop} = 1$ та $K_{mut} = 7$

Таблиця 1

Усереднене значення кількості сформованих S-блоків (K_{Sbox}) до знаходження S-блоку з $N_f = 104$

	K_{pop}										
K_{mut}	1	3	5	7	9	11	13	15	17	19	21
1	58 213	65 942	72 830	86 642	101 726	111 990	112 718	125 113	132 806	140 336	149 339
4	56 067	64 863	75 069	89 598	94 726	105 925	122 364	137 003	136 740	151 874	163 291
7	49 277	67 198	77 848	88 353	103 154	109 618	122 382	130 901	142 463	144 601	165 918
10	54 636	65 723	82 198	92 542	102 797	114 163	129 411	137 442	147 416	161 020	165 672
13	56 042	62 660	83 216	94 538	101 073	117 611	124 466	135 244	152 048	158 696	171 756
16	56 010	68 711	79 645	93 134	107 371	120 567	125 274	140 817	150 494	155 049	169 462
19	56 532	65 910	82 883	92 911	105 144	117 877	129 718	142 017	155 463	164 902	175 531
22	54 775	67 236	77 663	92 874	105 559	120 992	131 029	140 772	156 224	162 808	176 669
25	50 066	70 394	79 596	98 967	115 462	118 406	135 294	144 708	157 321	177 621	183 087
28	54 203	70 453	82 200	91 841	108 783	121 683	133 665	152 984	159 887	176 751	181 781
31	53 709	71 581	91 827	101 536	109 625	126 616	143 233	156 987	160 573	183 069	192 493

Таблиця 2

Усереднений час (секунди) до знаходження S-блоку з $N_f = 104$

	K_{pop}										
K_{mut}	1	3	5	7	9	11	13	15	17	19	21
1	64	23	17	16	15	18	16	20	19	19	42
4	66	28	17	11	9	14	14	20	18	19	30
7	57	32	21	10	9	14	13	18	18	17	28
10	42	33	25	10	9	15	14	19	19	19	24
13	44	30	26	10	8	15	13	19	20	19	22
16	44	32	24	10	9	15	14	19	19	18	20
19	56	35	24	13	8	15	14	19	19	19	19
22	84	34	21	21	9	15	14	19	19	19	18
25	46	26	18	17	16	15	14	19	20	21	18
28	62	24	19	15	14	15	14	21	20	21	18
31	69	30	20	15	14	16	15	21	20	21	18

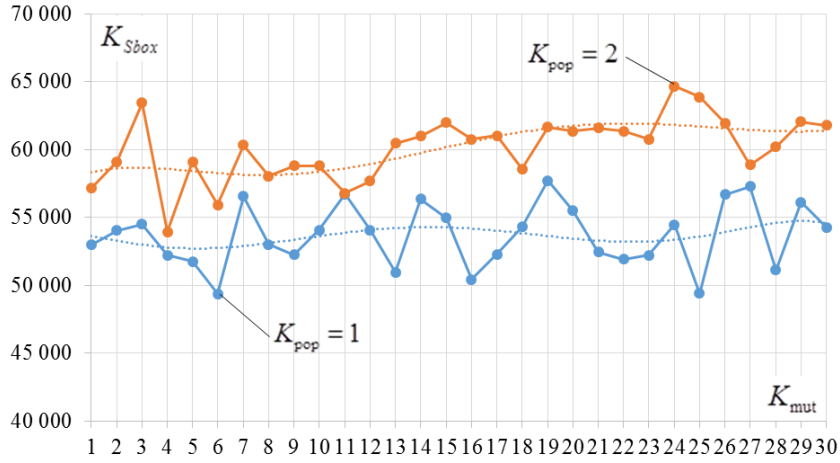


Рис. 7. Усереднене значення кількості сформованих S-блоків (K_{Sbox}) до знаходження S-блоку з $N_f = 104$ для K_{pop}=1 та 2

Таким чином, найменша кількість K_{Sbox} для алгоритму селекції становить близько 53 000 при $K_{pop} = 1$ та $K_{mut} = 4-9$. З ймовірністю 99 % під час пошуку буде знайдено S-блок з нелінійністю $N_f = 104$.

4. Обговорення результатів дослідження

4.1. Обговорення результатів

Отримані результати показують, що модифікований генетичний алгоритм з функцією WHS та оптимальними параметрами R=12 та X=0 є ефективним засобом для генерації S-boxes з високою нелінійністю. Використання селекції та додаткового зберігання поточної популяції дозволило значно покращити ефективність алгоритму порівняно з класичним генетичним алгоритмом.

Наші експерименти показали, що найкращі результати досягаються при використанні популяції з одного екземпляра та кількості мутацій у діапазоні від 4 до 9. У середньому, алгоритм вимагав перевірки близько 53 000 S-блоків для знаходження S-блоку з нелінійністю $N_f = 104$. Важливо зазначити, що при малих значеннях K_{pop} спостерігалися значні коливання від середнього значення, що свідчить про високу варіативність процесу пошуку.

Модифікований алгоритм селекції показав свою перевагу перед класичним генетичним алгоритмом. У більшості випадків він досягав мети з нелінійністю $N_f = 104$ з ймовірністю 99 %. Це підтверджує ефективність обраної стратегії селекції та підходу до мутацій, що забезпечують високий рівень стійкості до атак.

Розподіл частот нелінійності, отриманий у результаті наших експериментів, показав близьку відповідність до теоретичних очікувань. Це свідчить про правильність вибраних параметрів та налаштувань алгоритму. Додатково було виявлено, що середньоквадратичне відхилення результатів зменшується з ростом кількості мутацій, що також впливає на стабільність пошуку.

4.2. Порівняння з іншими методами

У порівнянні з іншими методами генерації S-boxes, такими як випадкова генерація або методи лінійних перетворень, наш підхід має кілька ключових переваг. Генетичний алгоритм з модифікованою селекцією дозволяє ефективно знаходити S-boxes з високими показниками нелінійності та стійкості до криптоаналізу. Крім того, він забезпечує гнучкість у налаштуванні параметрів, що дозволяє адаптувати алгоритм під конкретні вимоги криптографічних систем.

5. Висновки

У дослідженні розглянуто використання генетичних алгоритмів для генерації S-boxes з високою нелінійністю. Запропонований підхід базувався на використанні цільової функції WHS з оптимальними параметрами R=12 та X=0, що дозволило досягти високої ефективності у генерації бієктивних S-блоків.

Результати дослідження показали, що модифікований генетичний алгоритм з додатковим зберіганням поточної популяції та використанням селекції значно покращив ефективність пошуку. Найкращі результати були досягнуті при $K_{\text{pop}} = 1$ та $K_{\text{mut}} = 4 - 9$, де середня кількість перевірених S-блоків становила близько 53 000.

Запропонований алгоритм показав високу стабільність та ефективність у генерації S-boxes з нелінійністю $N_f = 104$, що підтверджується 99 % ймовірністю досягнення мети. Ці результати свідчать про перспективність використання генетичних алгоритмів у криптографічних застосуваннях, де потрібна висока стійкість до атак.

У майбутньому планується подальше вдосконалення алгоритму шляхом дослідження інших методів мутацій та селекції, а також оптимізація параметрів для досягнення ще кращих результатів. Крім того, можливе використання розподілених обчислень для подальшого прискорення процесу генерації S-boxes.

Список літератури:

1. Mishra N., Hafizul Islam S., Zeadally S. A survey on security and cryptographic perspective of Industrial-Internet-of-Things // Internet of Things. 2024. Vol. 25. P. 101037.
2. Urooj S. et al. Cryptographic Data Security for Reliable Wireless Sensor Network // Alexandria Engineering Journal. 2023. Vol. 72. P. 37–50.
3. Tiwari A. Chapter 14 – Cryptography in blockchain // Distributed Computing to Blockchain ; ed. Pandey R., Goundar S., Fatima S. Academic Press, 2023. P. 251–265.
4. A S C., S P M., R K P. Implementation of S-box for lightweight block cipher // 2023 3rd International Conference on Intelligent Technologies (CONIT). 2023. P. 1–4.
5. R M., V N.K. Optimized Implementation of S-box and Inverse S-box for PRESENT Lightweight Block Cipher // 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN). 2023. P. 1–5.
6. Teja P.R., Sasamal T.N. Implementation of Efficient Serial Architecture for Prince Block Cipher with Enhanced Security // 2023 International Conference on System, Computation, Automation and Networking (ICSCAN). 2023. P. 1–6.
7. Grami A. Chapter 11 – Cryptography // Discrete Mathematics ; ed. Grami A. Academic Press, 2023. P. 197–210.

8. Milanič M., Servatius B., Servatius H. Chapter 8 – Codes and cyphers // Discrete Mathematics With Logic / ed. Milanič M., Servatius B., Servatius H. Academic Press, 2024. P. 163–179.

9. McLaughlin J. Applications of search techniques to cryptanalysis and the construction of cipher components: phd. University of York, 2012.

10.Álvarez-Cubero J. Vector Boolean Functions: applications in symmetric cryptography. 2015.

11.Burnett L.D. Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography: phd. Queensland University of Technology, 2005.

12.Clark A.J. Optimisation heuristics for cryptology: phd. Queensland University of Technology, 1998.

13.Fuller J.E. Analysis of affine equivalent boolean functions for cryptography: phd. Queensland University of Technology, 2003.

14.Carlet C., Ding C. Nonlinearities of S-boxes // Finite Fields and Their Applications. 2007. Vol. 13, № 1. P. 121–135.

15.Ghosh A., Das S., Saha B. Chapter 6 - Nature-inspired optimization algorithms // Artificial Intelligence in Textile Engineering / ed. Ghosh A., Das S., Saha B. Woodhead Publishing, 2024. P. 171–231.

16.Tsai C.-W., Chiang M.-C. Chapter Seven - Genetic algorithm // Handbook of Metaheuristic Algorithms ; ed. Tsai C.-W., Chiang M.-C. Academic Press, 2023. P. 111–138.

17.Tsai C.-W., Chiang M.-C. Chapter Fifteen – Hybrid metaheuristic and hyperheuristic algorithms // Handbook of Metaheuristic Algorithms ; ed. Tsai C.-W., Chiang M.-C. Academic Press, 2023. P. 321–350.

18.Tesar P. A New Method for Generating High Non-linearity S-Boxes. Společnost pro radioelektronické inženýrství, 2010.

19.Ivanov G., Nikolov N., Nikova S. Reversed genetic algorithms for generation of bijective s-boxes with good cryptographic properties // Cryptogr. Commun. 2016. Vol. 8, № 2. P. 247–276.

20.Clark J.A., Jacob J.L., Stepney S. The design of s-boxes by simulated annealing // Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). 2004. Vol. 2. P. 1533–1537 Vol.2.

21.Kuznetsov A. et al. WHS Cost Function for Generating S-boxes // IEEE Int. Conf. Probl. Infocommunications, Sci. Technol., PIC S T – Proc. Institute of Electrical and Electronics Engineers Inc., 2021. P. 434–438.

22.Kuznetsov A. et al. Opportunities to minimize hardware and software costs for implementing boolean functions in stream ciphers // Int. J. Comput. Research Institute of Intelligent Computer Systems, 2019. Vol. 18, № 4. P. 443–452.

Надійшла до редколегії 08.06.2024

Відомості про авторів:

Кузнецов Олександр Олександрович – д-р техн. наук, професор, професор кафедри безпеки інформаційних систем і технологій; Харківський національний університет імені В. Н. Каразіна; Харків, Україна; e-mail: kuznetsov@karazin.ua, n.poluyanenko@karazin.ua; ORCID: https://orcid.org/0000-0003-2331-6326

Полуяненко Микола Олександрович – канд. техн. наук, доцент, доцент кафедри безпеки інформаційних систем і технологій; Харківський національний університет імені В. Н. Каразіна; Харків, Україна; e-mail: n.poluyanenko@karazin.ua; ORCID: https://orcid.org/0000-0001-9386-2547

Прокопович-Ткаченко Дмитро Ігорович – канд. техн. наук, доцент, доцент кафедри кібербезпеки та інформційних технологій; Університет митної справи та фінансів; Дніпро, Україна; e-mail: omega2417@gmail.com; ORCID: https://orcid.org/0000-0002-6590-3898

Котух Євген Володимирович – канд. техн. наук, професор кафедри кібербезпеки; Національний технічний університет «Дніпровська політехніка»; Дніпро, Україна; e-mail: yevgenkotukh@gmail.com; ORCID: https://orcid.org/0000-0003-4997-620X

Любчак Володимир Олександрович – канд. фіз.-мат. наук, доцент, завідувач кафедри кібербезпеки; Сумський державний університет; e-mail: v.liubchak@dcs.sumdu.edu.ua; ORCID: https://orcid.org/0000-0002-7335-6716