УДК 681.142.2; 622.02.658.284; 621. 325

Ю. Рашкевич, А. Ковальчук, Д. Пелешко, М. Купчак
Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління

ПОТОКОВА МОДИФІКАЦІЯ АЛГОРИТМУ RSA ДЛЯ ШИФРУВАННЯ ЗОБРАЖЕНЬ З ЧІТКО ВИДІЛЕНИМИ КОНТУРАМИ

На основі формування потоків даних зображення запропоновано модифікацію алгоритму RSA для шифрування зображень з однобайтовим піксельним форматом і таких, що дозволяють чітко виділяти контури.

There is proposed RSA algorithm modification on the based of the form of pixel threads. The used picture must be in the one byte pixel format and allows contour separation.

Вступ

Криптографія (від грецького kryptós — прихований і gráphein — писати) — наука про математичні методи забезпечення конфіденційності (неможливості прочитання інформації стороннім) і автентичності (цілісності і справжності авторства) інформації. Розвинулась з практичної потреби передавати якнайнадійніше важливі відомості. Для математичного аналізу криптографія використовує інструментарій абстрактної алгебри.

Для сучасної криптографії характерне використання відкритих алгоритмів шифрування, що допускають використання обчислювальних засобів. Відомо більше десятка перевірених алгоритмів шифрування, які, при використанні ключа достатньої довжини і коректної реалізації алгоритму, роблять шифрований текст недоступним для криптоаналізу. До таких алгоритмів належать **Twofish, IDEA, RC4** та ін [1].

Широкі академічні дослідження криптографії з'явились порівняно нещодавно — починаючи з середини 70-х, разом із появою відкритої специфікації стандарту **DES (Data Encryption Standard**) Національного бюро стандартів США (**National Bureau of Standards, NBS**), публікацій Діффі-Хелмана та публікації алгоритму **RSA**. Відтоді криптографія перетворилась на загальнопоширений інструмент для передавання даних у комп'ютерних мережах та захисту інформації загалом. Сучасний рівень безпеки багатьох криптографічних методів ґрунтується на складності деяких обчислювальних задач, таких як розклад цілих чисел, або проблеми з дискретними логарифмами. У багатьох випадках існують докази безпечності криптографічних методів лише за умови неможливості ефективного розв'язання певної обчислювальної задачі. Тут окремим винятком є шифрування за схемою одноразових блокнотів.

У багатьох країнах прийнято національні стандарти шифрування. У 2001 році в США прийнятий стандарт симетричного шифрування **AES** на основі алгоритму Rijndael з довжинами ключів 128, 192 і 256 біт. Алгоритм **AES** прийшов на зміну колишньому алгоритмові DES, який тепер рекомендовано використовувати тільки в режимі **Triple-DES (3DES)**.

I. Огляд основних алгоритмів шифрування

1. Симетричні алгоритми шифрування

Шифри **Data Encryption Standard (DES)** та **Advanced Encryption Standard (AES)** є стандартами блочних шифрів, затверджених урядом США (стандартизацію **DES** скасовано після прийняття стандарту **AES**). Незважаючи на те, що стандарт **DES** визнано застарілим, він (а особливо його все ще дійсний варіант **Triple-DES**) залишається достатньо популярним і використовується в багатьох випадках – від шифрування в банкоматах до забезпечення приватності електронного листування та безпечного доступу до віддалених терміналів. Було також розроблено багато інших шифрів різної якості:

•**DES (англ. Data Encryption Standard)** – це симетричний алгоритм шифрування даних в основу якого був покладений шифр Хорста Фейстеля Люцифер. Цей стандарт шифрування прийнятий урядом США і в період із 1976 до кінця 1990-х набув міжнародного поширення. Ще з часу свого розроблення алгоритм викликав неоднозначні відгуки. Оскільки **DES** містив засекречені елементи своєї структури, виникали побоювання щодо можливості контролю з боку Національного Агентства Безпеки США **(National Security Agency)**. Алгоритм піддавався критиці за малу довжину ключа. Проте це не завадило йому стати загальноприйнятим стандартом. DES був основою для сучасних уявлень про блочні алгоритми шифрування та криптоаналіз.

Оскільки **DES** – порівняно старий криптоалгоритм, існує багато публікацій щодо його криптоаналізу. Ґрунтовну оцінку безпеки **DES** дає Брюс Шнаєр, який у [2] детально розбирає та впорядковує велику кількість публікацій щодо криптоаналізу **DES**. Сьогодні **DES** вважається нестійким, оскільки:

1) розмір ключа – 56 бітів – замалий, тому існує реальна загроза пошуку ключа “лобовою” атакою (послідовним перебором).

2) **DES** нестійкий до лінійного криптоаналізу (тобто лінійна атака дає змогу знайти ключ **DES** швидше, ніж послідовний перебір).

Водночас, повний 16-раундовий **DES** стійкий до диференційного криптоаналізу.

Через велику поширеність **DES** було запропоновано багато способів підвищення його безпеки, зокрема, замінити S-блоки **DES** новими, які є стійкішими до лінійної атаки. Однак, широкого практичного застосування жодна з видозмінених версій **DES** не набула. Винятком є **3DES**, однак, це не видозміна алгоритму, а лише особливий режим шифрування звичайним **DES**.

•**Advanced Encryption Standard (AES)**, також відомий під назвою **Rijndael** — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт) прийнятий як американский стандарт шифрування урядом США у 2002 році. Вибір припав на **AES**, зважаючи на широке використання і активний аналіз алгоритму. Станом на 2006 рік **AES** є одним із найпоширеніших алгоритмів симетричного шифрування.

Потреба у новому стандарті шифрування виникла у середині 90-х років. Існуючий стандарт DES з довжиною ключа 56 бітів, дав змогу застосувати метод “лобової” атаки для дешифрування даних. Крім того, архітектура **DES** орієнтувалась на апаратну реалізацію, а програмна реалізація на платформах із обмеженими ресурсами не забезпечувала необхідної швидкості застосування. Модифікація **DES, 3-DES** мала достатню довжину ключа, але при цьому була ще повільнішою.

Алгоритм **Rijndael** підтримує широкий діапазон розміру блоку та ключа. **AES** має фіксовану довжину у 128 бітів, а розмір ключа може набувати значень 128, 192 або 256 бітів, тоді як Рейндол підтримує розмірність блоку та ключа із кроком 32 біт у діапазоні від 128 до 256. Через фіксований розмір блоку **AES** оперує із масивом 4×4 байт, що називається станом (версії алгоритму із більшим розміром блоку мають додаткові колонки).

Потокові шифри, на відміну від блочних, створюють ключ довільної довжини, що накладається на відкритий текст побітово або політерно, і є подібними до одноразового блокноту. В потокових шифрах потік шифротексту обчислюється на основі внутрішнього стану алгоритму, який змінюється протягом його дії. Зміна стану керується ключем, та в деяких алгоритмах ще і потоком відкритого тексту. **RC4** є прикладом добре відомого та поширеного потокового шифру [8].

Криптографічні хешувальні функції (cryptographic hash functions, або message digest functions) не обов'язково використовують ключі, але часто використовуються і є важливим класом криптографічних алгоритмів. Ці функції отримують дані (часто, ціле повідомлення), та обчислюють коротке, фіксованого розміру число (хеш). Добрі хешувальні функції створені так, що дуже важко знайти колізії (два відкриті тексти, що мають однакове значення хешу).

Коди аутентифікації повідомлень (message authentication code, **MAC**) подібні до криптографічних хешувальних функцій, за винятком того, що вони використовують секретний ключ для аутентифікації значення хешу при отриманні повідомлення. Ці функції пропонують захист проти атак на прості хешувальні функції.

Блочний шифр – різновид симетричного шифру. Особливістю блочного шифру є обробка блоку декількох байтів за одну ітерацію (як правило, 8 або 16).

Робота блочного шифру в найпростішому режимі – застосування шифрувальної функції до блоку даних (проста заміна) викликає серйозну проблему: статистичні властивості відкритих даних частково зберігаються, тому що кожному однаковому блоку даних однозначно відповідає зашифрований блок даних. За великої кількості даних (відео, звук) це може дати деякі відомості для криптоаналізу про зміст даних.

Видалення статистичних залежностей у відкритому тексті можливе за допомогою попереднього архівування, але воно не вирішує завдання повністю, тому що у файлі залишається службова інформація архіватора, і це не завжди є допустимим. Для вирішення вищеописаних проблем використовується циклічне кодування ("категоризація") даних. Зміст цього рішення полягає в накладанні статистичних даних у тривимірному просторі й розподілу рейтингу кожній групі даних. У результаті цього можна вилучати ділянки з низьким рейтингом і так розділити кодовані ділянки від "порожніх" комбінацій.

2. Асиметричне шифрування

Асиметричні алгоритми шифрування — алгоритми шифрування, які використовують різні ключі для шифрування та дешифрування даних.

Основною перевагою асиметричного шифрування є забезпечення обміну секретними повідомленнями між агентами, які не мають попередньої домовленості про безпеку. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналу зникає.

Прикладами криптосистем з відкритим ключем є **Elgamal** (автор Тахір Ельгамаль), **RSA** (автори: Рон Рівест, Аді Шамір і Леонардо Адлман), **Diffie-Hellman і DSA, Digital Signature Algorithm** (автор Девід Кравіц).

Проблему керування ключами вирішила криптографія з відкритим, або асиметричним, ключем, концепцію якої запропонували Уітфілд Діффі і Мартін Хеллман у 1975 році. Криптографія з відкритим ключем – це асиметрична схема, у якій застосовуються пари ключів: відкритий (**public key**), який зашифровує дані, і відповідний йому закритий (**private key**), що їх розшифровує. Відкритий ключ, на відміну від закритого, вільно поширюється. За його допомогою будь-який агент може зашифрувати інформацію, яка розшифровується винятково закритим ключем.

Незважаючи на те, що пара ключів є математично зв'язаною, обчислития закритий ключ за допомогою відкритого практично неможливо.

Поява шифрування з відкритим ключем стала технологічною революцією, яка зробила стійку криптографію доступною в широкому використанні. Коротко розглянемо базовий алгоритм асиметричного шифрування.

• **RSA** — криптографічна система з відкритим ключем. **RSA** став першим алгоритмом такого типу, придатним і для шифрування, і для цифрового підпису. Алгоритм використовується у великій кількості криптографічних додатків.

RSA опубліковано у 1977 році Рональдом Райвестом (Ronald Linn Rivest), Аді Шаміром (**Adi Shamir**) і Леонардом Адлеманом (**Leonard Adleman**) з Масачуссетського технологічного інституту (**MIT**).

В 1983 році виданий патент 4405829 США, термін дії якого минув 21 вересня 2000 року.

Безпека алгоритму **RSA** побудована за принципом складності факторизації. Алгоритм використовує два ключі – відкритий і закритий.

Для того, щоб згенерувати пари ключів, виконуються такі дії:

1) вибираються два великих простих числа z_1 і z_2;
2) обчислюється їх добуток

$$z = z_1 z_2; \quad (1)$$

3) обчислюється функція Ейлера

$$\varphi(z) = (z_1 - 1)(z_2 - 1); \quad (2)$$

4) вибирається ціле e таке, що $1 < e < \varphi(z)$ та e і $\varphi(z)$ є взаємно простими.

5) з допомогою розширеного алгоритму Евкліда знаходиться число d таке, що,

$$ed \equiv 1\left(\mathrm{mod}\left(\varphi(z)\right)\right). \quad (3)$$

Число z називається модулем, а числа e і d – відкритою й закритою (секретною) експонентами, відповідно. Пара чисел (z, e) є відкритою частиною, а d – секретною частиною ключа. Числа z_1 і z_2 після генерації пари ключів можуть бути знищені, але в жодному разі не повинні бути розкриті.

Для того, щоб зашифрувати повідомлення $c_{i,j} < z$, обчислюється

$$c'_{i,j} = c_{i,j}^{e} \bmod z. \quad (4)$$

Число $c'_{i,j}$ використовується як шифротекст. Для розшифрування потрібно обчислити

$$c_{i,j} = c'^{d}_{i,j} \bmod z. \quad (5)$$

Неважко переконатися, що при розшифруванні ми відновимо вихідне повідомлення:

$$c'^{d}_{i,j} \equiv \left(c_{i,j}^{e}\right)^{d} \equiv c_{i,j}^{ed} \left(\mathrm{mod}\, z\right). \quad (6)$$

З умови (3) випливає, що

$$ed = k\varphi(z)+1, \quad (7)$$

для деякого цілого k, отже

$$c_{i,j}^{ed} \equiv c_{i,j}^{k\varphi(z)+1} \left(\mathrm{mod}\, z\right). \quad (8)$$

Згідно з теоремою Ейлера:

$$c_{i,j}^{\varphi(z)} \equiv 1\left(\mathrm{mod}\, z\right), \quad (9)$$

тому

$$c_{i,j}^{k\varphi(z)+1} \equiv c_{i,j}^{k\varphi(z)+1} \left(\mathrm{mod}\, z\right) \;\rightarrow\; c'^{d}_{i,j} \equiv c_{i,j} \left(\mathrm{mod}\, z\right), \quad (10)$$

- **Цифровий підпис. RSA** може використовуватися не тільки для шифрування, але й для цифрового підпису. Підпис s повідомлення m обчислюється з використанням секретного ключа за формулою:

$$s = m^{d} \bmod z. \quad (11)$$

Для перевірки правильності підпису потрібно переконатися, що виконується рівність

$$m = s^{e} \bmod z. \quad (12)$$

Деякі особливості алгоритму

Для знаходження двох великих простих чисел z_1 і z_2 при генерації ключа зазвичай використовуються ймовірнісні тести чисел на простоту, які дають змогу швидко виявити й відкинути складні числа.

Для генерації z_1 і z_2 необхідно використовувати криптографічно надійний генератор випадкових чисел. При цьому z_1 і z_2 не повинні бути занадто близькими, оскільки в цьому випадку завдяки методу факторизації Ферма їх можна буде віднайти. Крім того, необхідно вибирати «сильні» прості числа, щоб уникнути «взлому» z_1 -1 алгоритмом Поларда. При практичному використанні необхідно певним чином доповнювати повідомлення. Відсутність доповнень може призвести до деяких проблем:

- значення $m = 0$ і $m = 1$ дадуть при шифруванні повідомлення 0 і 1 при будь-яких значеннях e і z.
- при малому значенні коефіцієнта (e = 3, наприклад) можлива ситуація, коли легко можна відновити вихідне повідомлення.
- оскільки RSA є детермінованим алгоритмом, тобто не використовує випадкових значень у процесі роботи, то може використати атаку з обраним відкритим текстом.

.

Для розв'язання цих проблем перед кожним шифруванням повідомлення доповнюються деяким випадковим значенням. Доповнення здійснюють так, щоб гарантувати, що $m \neq 0$, $m \neq 1$ і $m^e > z$. Крім того, оскільки повідомлення доповнюється випадковими даними, то, шифруючи той самий відкритий текст, щораз отримуватимуть інше значення шифртексту, що робить атаку з обраним відкритим текстом неможливою.

RSA працює значно повільніше, ніж симетричні алгоритми. Для підвищення швидкості шифрування e вибирається невеликим, зазвичай 3, 17 або 65537. Ці числа у двійковому вигляді містять тільки по дві одиниці, що зменшує кількість необхідних операцій множення при піднесенні до степеня. Наприклад, для піднесення кількісті до степеня 17 потрібно виконати тільки 5 операцій множення.

Вибір малого значення e може призвести до розкриття повідомлення, якщо воно відправляється відразу декільком адресатам. Проте ця проблема вирішується організацією доповнення повідомлень.

Значення секретного коефіцієнта d повинне бути достатньо великим. М. Вінер (**Michael J. Wiener**) показав, що якщо $z_2 < z_1 < 2z_2$ і $d < z^{\frac{1}{4}}/3$, то існує ефективний спосіб обчислити d по z і e. Однак, якщо значення e вибирається невеликим, то d отримується достатньо великим, і проблеми нівелюється.

Система **RSA** використовується для захисту програмного забезпечення й у схемах цифрового підпису. Також вона використовується у відкритій системі шифрування **PGP**.

Через низьку швидкість шифрування (близько 30 кбіт/сек при 512-бітному ключі на процесорі 2 ГГц) повідомлення зазвичай шифрують за допомогою продуктивніших симетричних алгоритмів з випадковим ключем (сеансовий ключ), а за допомогою **RSA** шифрують лише цей ключ.

Характеристика об’єкта дослідження

Об’єктом дослідження обрано алгоритм шифрування **RSA**. Метою **RSA** є стійке до несанкціонованого доступу шифрування сигналів. Як сигнал у подальших дослідженнях обрано зображення з однобайтовим піксельним форматом [2] і такі, які дають змогу дуже якісно виділяти контури.

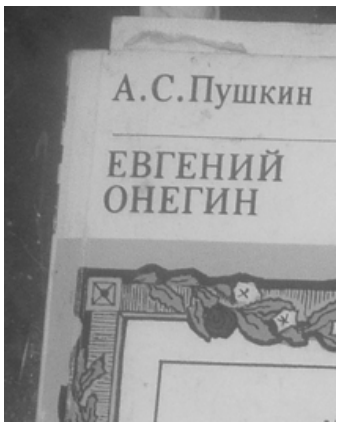

Рис.1. Оригінальне зображення

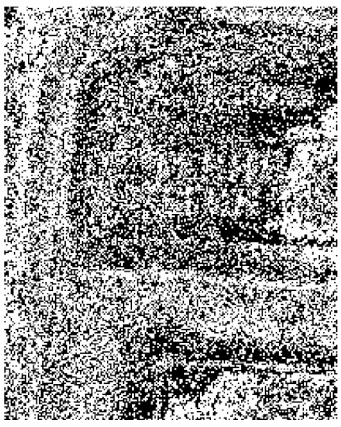

Рис.2. Зашифроване алгоритмом RSA зображення за таких параметрів: $z_2 = 47$; $z_2 = 71$; $e = 79$

Використання алгоритму шифрування **RSA** [1] як найстійкішого до несанкціонованого дешифрування кодованих сигналів, стосовно зображень, які дають змогу дуже строго виділяти контури, не дає задовільних результатів. Це добре ілюструють приклади, наведені на рис.1, 2. На рис. 2 наведено зашифроване зображення, оригінал якого наведено на рис. 1. Як можна побачити з рис.2,

зашифроване зображення є ”шумовим” набором попіксельних значень кольорів. Проте на ньому все ж таки проступають основні контури вхідного зображення, тобто має місце ефект неповного зашумлення зображення.

Цей ефект описаний у [4] і зумовлений логікою математичного перетворення – оператора шифрування алгоритму **RSA** [1].

Отримане в результаті шифрування за алгоритмом **RSA** зображення не можна вважати повністю зашифрованим. Це пояснюється тим, що несанкціоноване дешифрування такого зображення, окрім традиційного “взлому” самого алгоритму (сьогодні авторам невідомий ефективний спосіб “взлому” алгоритму **RSA**), допускає операції над цим зображенням традиційними засобами обробки зображень, зокрема фільтрації, реконструкції і ін. Незважаючи на те, що другий спосіб не дає повного відтворення вхідного зображення, за його допомогою не санкціоновано отримати певну інформативність із зашифрованого зображення видається цілком можливим.

Постановка задачі

Основним завданням статті є потокова модифікація алгоритму **RSA** для випадку використання його стосовно зображень, які дають змогу дуже чітко виділяти контури. Це дасть можливість уникнути ефекту появи контурів у зашифрованих зображеннях і тим самим нівелювати несанкціоноване відтворення зашифрованого зображення.

Схема алгоритму

Нехай задано зображення P розмірностей h – висота – кількість пікселів по вертикалі, та l – довжина – кількість пікселів по горизонталі. Це зображення можна розглядати як матрицю пікселів $\mathbf{P}_{l,h}$, відповідною до якої є матриця кольорів $\mathbf{C}_{\mathbf{P}_{l,h}}$

$$P = \mathbf{P}_{l,h} = \left[pxl_{ij} \right]_{1 \le i \le n(l), 1 \le j \le m(h)} \to \mathbf{C}_{\mathbf{P}_{l,h}} = \left[c_{ij} \right]_{1 \le i \le n(l), 1 \le j \le m(h)}, \quad (13)$$

де pxl_{ij} – піксел з кординатами (i, j) дискретизованого зображення P; n та m – кількості пікселів у напрямках l та h відповідно.

Сформуємо потік кольорів шляхом склеювання векторів кольорів усіх рядків рисунка. Довжина потоку дорівнює $n \times m$.

Надалі сформуємо розбиття потоку на відрізки C_i завдовжки α. Шифрування здійснюватимемо за схемою зчеплення блоків шифру (рис. 3).

Шифрування відбувається такими етапами:

1. Перший блок C_0 сумується за модулем із випадковим вектором ініціалізації. Вектор ініціалізації використовується для ускладнення викриття шифру. Оскільки RSA має достатню криптостійкість, то вектор ініціалізації можна не враховувати.

2. Зашифруємо блок C_i за алгоритмом RSA.

3. Для кожного блоку $\boldsymbol{C}_{i+1}$, підсумовуємо його за модулем 2 із першими α байтами зашифрованої послідовності $\boldsymbol{C}'_i$, отриманої на попередньому кроці.

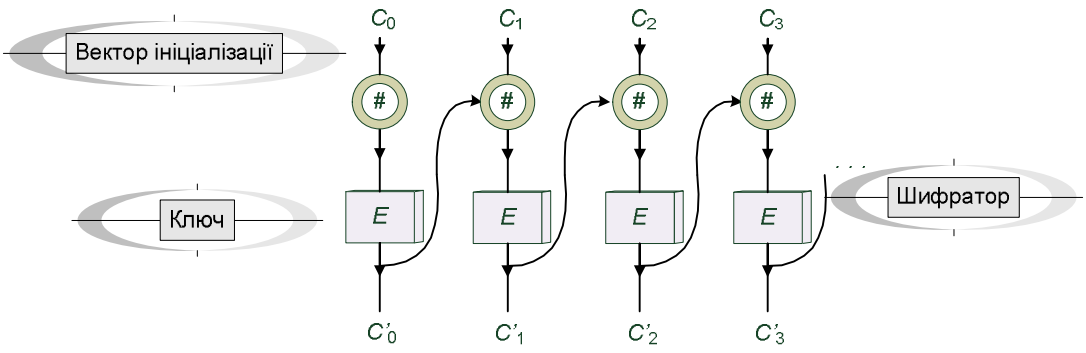

Рис.3. Схема шифрування

Дешифрування здійснюватимемо за схемою, наведеною на рис. 4.

Дешифрування відбувається такими етапами:

1. Розшифруємо блок $\boldsymbol{C}'_0$ і підсумуємо результат за модулем 2 із вектором ініціалізації.
2. Кожен наступний блок $\boldsymbol{C}'_{i+1}$ розшифруємо за алгоритмом **RSA** і підсумуємо за модулем 2 із першими α байтами зашифрованого блоку $\boldsymbol{C}_i$.

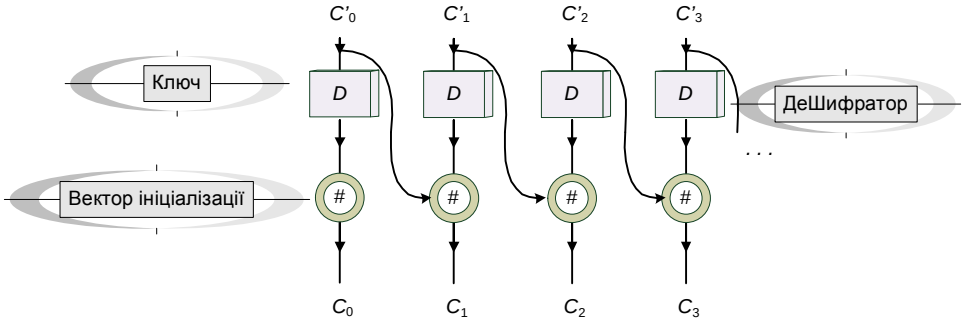

Рис.4. Схема дешифрування

Програмна реалізація

Головне вікно програми (рис. 5) містить головне меню, через яке здійснюються операції завантаження рисунків та шифрування/дешифрування, список завантажених зображень та області із поточними та опрацьованими зображеннями.

На рис. 5 показано головне вікно програми після шифрування тестового зображення, а на рис. 6 – після дешифрування.

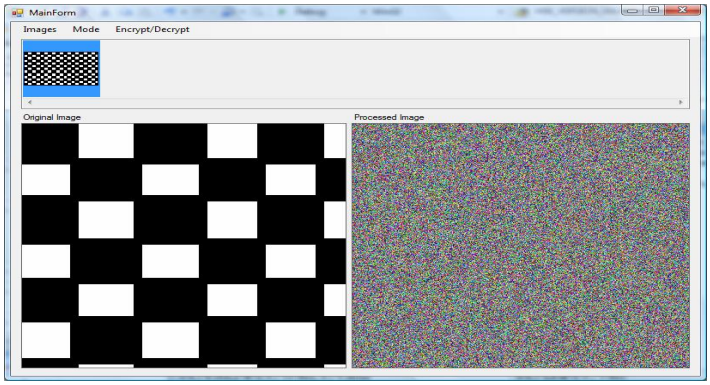

Рис. 5. Результати шифрування зображення

Діаграма класів наведена на рис. 7. Основним класом є BlockAlgorithm, який реалізує базовий інтерфейс для алгоритмів. Цей клас є абстрактним функтором, який вимагає перевантаження методу do_process.

```
ref class BlockAlgorithm abstract
{
        protected:
          virtual ImageStream ^ do_process(ImageStream ^ data) { return gcnew ImageStream; };
          array<System::Byte> ^ initialVector_;

        public:
          BlockAlgorithm(array<System::Byte> ^ initialVector): initialVector_(initialVector) {};
          virtual ~BlockAlgorithm(void) {};

          ImageStream ^ operator()(ImageStream ^ data) {
            return do_process(data);
  };
};
```

Від класу BlockAlgorithm наслідується RSABlockCoder та RSABlockDecoder – класи, які реалізують віртуальну функцію do_process для блочного шифрування та дешифрування відповідно.

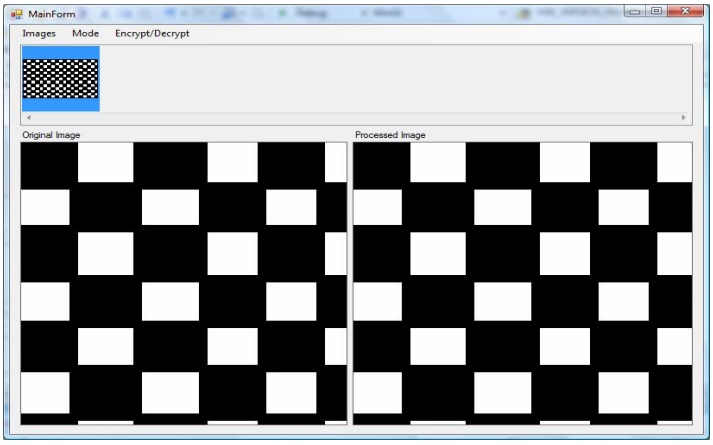

Рис. 6 . Результати дешифрування зображення

Сигнатура методу **do_process** є такою:
virtual ImageStream ^ do_process(ImageStream ^ data) { return gcnew ImageStream; };

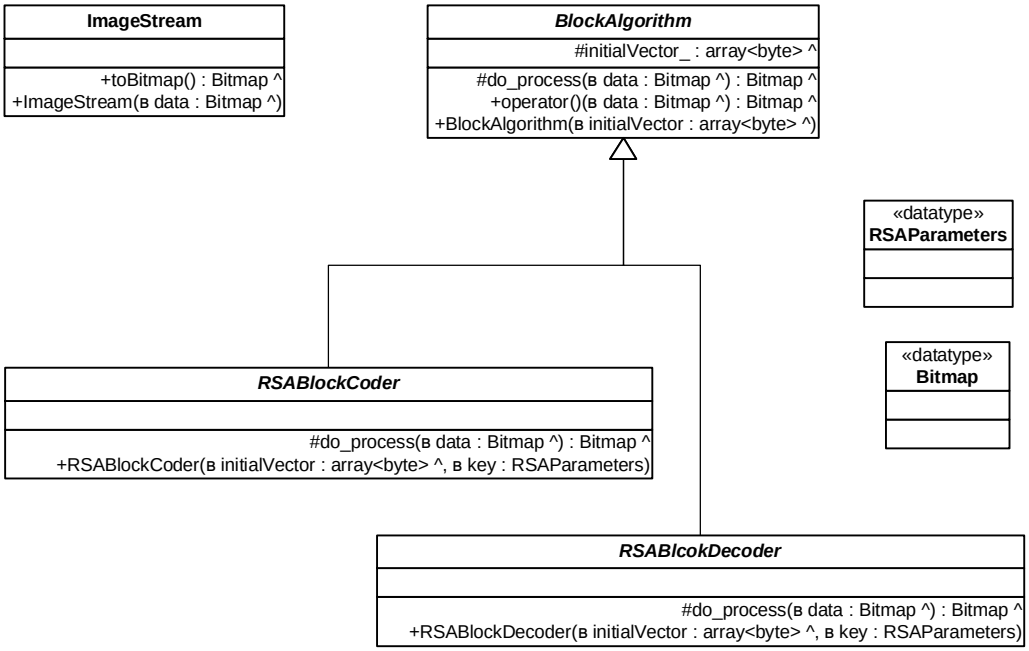

Рис. 7. Діаграма класів

Наведений на рис.7 клас **ImageStream** є допоміжним класом, який перетворює рисунок на потік байтів.

У просторі імен **BlockRSA**, реалізовані дві функції **Encrypt** і **Decrypt**, які виконують ініціалізацію об’єктів та здійснюють шифрування та дешифрування відповідно.

Висновки

Стійкість до несанкціонованого дешифрування запропонованого алгоритму повністю відповідає стійкості алгоритму **RSA**. Проте, як показали результати практичного використання описа-

ної модифікації **RSA**, у зашифрованому зображенні вдалось досягти повної зашумленості і уникнути областей однорідності (рис.5). При цьому виявити контури вихідного зображення стає практично неможливим. Це дало можливість нівелювати можливості використання традиційних засобів обробки зображень для несанкціонованого отримання інформації із зашифрованого зображення.

Описаний алгоритм може бути застосований для будь-яких зображень. В усіх випадках буде отримано результат повного зашумлення зображення.

1. Компьютерные сети. 4-е изд. / Е. Тененбаум. – СПб.: Питер, 2006. – С. 832-852. 2. Шнайдер Б Прикладная криптография.– М.: Издательство Триумф, 2003. – 816 с. 3. Гонсалес Р., Вудс Р. Цифровая обработка изображений.– М.: Техносфера, 2005. – 1072 с. 4. Рашкевич Ю.М., Пелешко Д.Д., Ковальчук А.М., Пелешко М.З. Модифікація алгоритму RSA для деяких класів зображень // Технічні вісті. – 2008/1(27), 2(28). – С. 59 – 62.

УДК 004.032.26:004.048

Р. Ткаченко, А. Дорошенко
Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління

НЕЙРОПОДІБНІ СТРУКТУРИ МАШИНИ ГЕОМЕТРИЧНИХ ПЕРЕТВОРЕНЬ У ЗАВДАННЯХ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

Показано особливості архітектури та застосування нейроподібних структур машини геометричних перетворень (МГП) до розв'язання задач передбачення в галузі інтелектуального аналізу даних (ІАД). Розроблений в статті метод передбачення на основі незалежного відтворення головних компонентів ілюструється прикладом.

The article describes the features of the architecture of neural networks and the approach to solving a problem of prediction for Data Mining tasks where data are high-dimensional. Essential principles of the methods of prediction on the base of neural networks base on geometrical transformation machine are proposed. This method of prediction is improved by independent reproduction of principal components.

Вступ

Розвиток технічних засобів та сучасних інформаційних технологій відбору призводить до накопичення величезних інформаційних масивів, які вимагають подальшого опрацювання та аналізу з метою ефективного використання даних в системах підтримки прийняття рішень. Велика розмірність завдань в галузі ІАД, істотні нелінійності та невизначеності різноманітної природи, виродженість задач, суперечливість, неповнота та неоднорідність даних передбачають в більшості випадків для отримання необхідних результатів застосування сучасних засобів штучного інтелекту, зокрема штучних нейронних мереж (ШНМ) та контролерів нечіткої логіки [1]. Існує також проблема інформаційного моделювання в ІАД, пов'язана з наявністю як глобальних залежностей між досліджуваними змінними, що діють однотипно в усій області визначення, так і локальних. Для моделювання і передбачення глобальних залежностей найчастіше застосовують багатошарові перцептрони, а для локальних – нейромережі радіальних базових функцій (РБФ) [2,7,8]. Отже, вказані вище нейромережні технології по суті виключають одна одну. Оскільки в багатьох реальних випадках структура даних є складнішою, відображає залежності обох типів, названий вище підхід