УДК 004.056.55 + 004.415.3 **doi: 10.26906/SUNZ.2025.4.45-50**

М. І. Главчев, Ю. М. Главчева, Г. І. Молчанов

Національний технічний університет «Харківський політехнічний інститут», Харків, Україна

РЕЖИМ БЛОЧНОГО ШИФРУВАННЯ НА ОСНОВІ КОДУ ХАФФМАНА ДЛЯ НІББЛІВ

Анотація. Актуальність. Актуальність роботи зумовлена потребою у криптографічних рішеннях, які поєднують високу безпеку, ефективність стиснення та низькі обчислювальні витрати, що особливо важливо для IoT, мобільних та вбудованих систем. **Метою статті** є дослідження розробки та аналіз режиму блочного шифрування на основі коду Хаффмана для ніббліів, з інтегрованими механізмами хеш-захисту та підтримкою як глобальної, так і пер-блокової обробки. **Об'єкт дослідження**: використання перетворення Хаффмана полубайтів (ніббліів) для створення режиму блокового шифрування. **Результати дослідження.** Проведені експерименти показали, що для даних із низькою ентропією ніббліів (H<3.5 біт/ніббл) запропонований режим зменшує обсяг зашифрованих даних на 10–20 % без суттєвої втрати швидкодії, забезпечуючи стійкість до помилок і збереження цілісності. **Висновок**: інтеграція статистичного кодування та симетричного шифрування з адаптивним вибором параметрів дозволяє отримати збалансоване рішення для безпечної та ефективної передачі інформації в умовах обмежених ресурсів.

Ключові слова: блочне шифрування, код Хаффмана, стиснення даних, криптографія, симетричні алгоритми.

Вступ

Постановка проблеми. Зі зростанням обсягів цифрового трафіку та поширенням мобільних і вбудованих пристроїв з обмеженими обчислювальними ресурсами постає необхідність у створенні криптографічних алгоритмів, які забезпечують належний рівень безпеки при мінімальних затратах пам'яті та процесорного часу. Традиційні блочні шифри, такі як AES чи DES, орієнтовані на роботу з блоками фіксованої довжини та не враховують статистичних особливостей вхідних даних, що призводить до збільшення обсягу переданої інформації та може створювати передбачувані патерни у шифротексті.

Використання коду Хаффмана у процесі шифрування дозволяє динамічно змінювати довжину кодових слів відповідно до частоти появи ніббліів, зменшуючи надлишковість і водночас підвищуючи складність криптоаналізу [1]. Проте інтеграція змінної довжини коду в блочну структуру вимагає вирішення проблеми синхронізації та забезпечення стійкості до атак, зокрема статистичних і диференціальних. Додатково важливим завданням є впровадження механізмів хеш-захисту (ХЗ) для перевірки цілісності та автентичності зашифрованих даних, що особливо актуально в умовах потенційного пошкодження чи підміни пакетів у каналах зв'язку. Таким чином, актуальною є розробка та дослідження режиму блочного шифрування, який поєднує адаптивне статистичне кодування, хешування та симетричне шифрування для 4-бітних блоків.

Аналіз останніх досліджень і публікацій. Використання алгоритмів статистичного кодування, зокрема коду Хаффмана, у криптографії досліджувалося у ряді наукових робіт. У [2] розглянуто інтеграцію кодування змінної довжини у блочні шифри з метою зменшення обсягу переданих даних, проте автори відзначають проблему синхронізації при пошкодженні бітового потоку. У [3] детально обговорюється архітектура безпеки в IoT для середовища пристроїв з обмеженими ресурсами, у тому числі використання блокових шифрів, хеш-функцій, потокових шифрів, високопродуктивних систем та пристроїв з низькими ресурсами для середовища IoT. Дослідження [4] аналізує використання ніббліів як базових одиниць обробки для легковагових шифрів у системах IoT, наголошуючи на їх продуктивності та знижених вимогах до пам'яті.

Водночас у [5] відзначено, що використання змінної довжини кодів без додаткового шифрування та контролю цілісності не гарантує належного рівня безпеки, оскільки частотний аналіз кодових слів може призвести до відновлення оригінальної інформації. Робота [6] розглядає впровадження хеш-функцій (SHA-256, SHA-3) у симетричні схеми шифрування як механізм запобігання атакам на цілісність. Для виявлення модифікацій автори [7] зосереджуємося на проблемі ідентифікації зразків у наборі, які не відповідають структурованим шаблонам, на основі використання методу хешування. Крім того, дослідження [8–10] демонструють перспективність комбінованих підходів, де стиснення, шифрування і хешування інтегруються в єдиний процес, оптимізуючи обчислювальні витрати та підвищуючи захист.

Метою роботи є створення та аналіз режиму блочного шифрування, який використовує код Хаффмана для оптимізації представлення ніббліів і підвищення стійкості до криптоаналітичних атак.

1. Кодування Хаффмана для ніббліів

Запропонований режим блочного шифрування ґрунтується на комбінації симетричного перетворення та адаптивного статистичного кодування за алгоритмом Хаффмана для 4-бітних блоків (ніббліів).

1. *Вхідні дані*. Послідовність відкритого тексту $M=\{m_1,m_2,\ldots,m_n\}$, де кожен елемент m_i - ніббл ($m_i \in \{0,1,\ldots,15\}$).

2. *Частотний аналіз*. Для множини ніббліів визначається частота появи: $p(m_i)=count(m_i)/n$, де

$$\sum_{i=0}^{15} p(m_i) = 1.$$

3. *Побудова кодового дерева Хаффмана*. Формується бінарне дерево, у якому кожному ніббліу

відповідає кодове слово змінної довжини l_i, таке що:

$$\sum_{i=0}^{15} p(m_i) - l_i \rightarrow min.$$

4. *Шифрування.* Отримані кодові слова пропускаються через симетричний блочний шифр:

$$C=E_K(H(M)),$$

де *H(M)* — закодоване Хаффманом повідомлення, E_K - блочне шифрування з ключем K.

5. *Хеш-захист*. Для перевірки цілісності формується хеш: *h=Hash(C)***.** У кінцевий пакет передаються (C,h).

Така модель дозволяє інтегрувати стиснення, шифрування та автентифікацію в єдиний процес, зменшуючи обсяг переданих даних і підвищуючи безпеку (рис. 1).

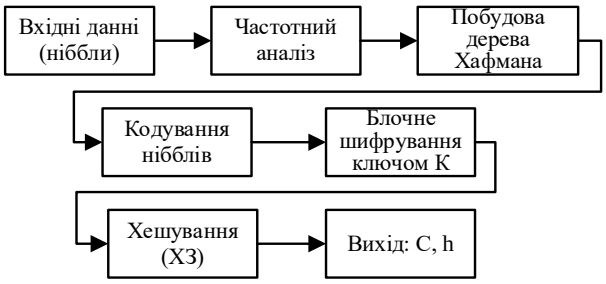

Рис. 1. Схема алгоритму блочного шифрування на основі коду Хаффмана для нібблів

Запропонована модель може бути застосована у таких напрямках:

1. Інтернет речей (IoT) — передавання телеметричних даних з датчиків, де важливі енергоефективність і зменшений обсяг пакету.

2. Мобільні додатки — захищене зберігання або передавання компактних даних (наприклад, токенів доступу, налаштувань).

3. Мультимедійні системи — стиснення та шифрування метаданих відео/аудіо потоків.

4. Вбудовані системи — криптографія для пристроїв з малим обсягом оперативної пам'яті та невисокою частотою процесора.

5. Фінансові сервіси — передача мікроплатіжних транзакцій з мінімізацією затримок і навантаження на канал.

У контексті запропонованого режиму використання нібблів забезпечує баланс між компактністю кодування та швидкістю обробки, особливо при інтеграції з кодом Хаффмана (табл. 1).

2. Послідовність режиму та шифру

Використання запропонованого режиму треба оцінити з можливість використання «до» та «після» шифрування (рис. 2). Розглянемо кожний з підходів.

1. Створення коду для нібблів на відкритому тексті, стиснення та потім шифрування.

1.1. *Ефективність.* Очікувана довжина коду:

$$E[L] = \sum_{i=0}^{15} p_i l_i ,$$

Де коефіцієнт стиснення *R = E[L] \ 4* (біт/ніббіт).

Таблиця 1 – **Порівняння байтів та нібблів**

Параметр	Байти (8 біт)	Ніббли (4 біти)
Кількість можливих значень	28=256	24=16
Розмір блока даних	Більший, вимагає більше пам'яті для обробки	Менший, зручний для ресурсно-обмежених пристроїв
Продуктивність	Вище у системах з широкими шинами даних	Вище у вузькоспеціалізованих легковагових шифрах
Гнучкість при кодуванні	Менша ефективність змінної довжини коду	Оптимальніше поєднується з кодом Хаффмана
Енергоспоживання	Вище при обробці малих обсягів даних	Нижче, що важливо для IoT
Криптостійкість	Більший простір ключів при однаковій кількості блоків	Потребує додаткових перетворень для стійкості

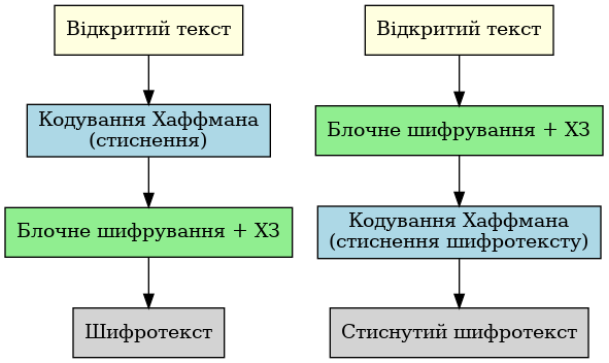

Рис. 2. Порівняння підходів застосування кодування Хаффмана

Якщо p_i нерівномірні, *R<1* ⇒ реальна економія. Для рівномірних — виграш мінімальний.

1.2. *Безпека.*

• Плюс: після шифрування бітовий потік стає рівномірним — складно робити частотний аналіз.

• Мінус/метадані: довжина шифротексту корелює з ентропією відкритого тексту (side channel по довжині).

Мітігації: паддинг до фіксованих рамок, фреймінг по чанках, AEAD (напр., AES GCM), «ключовий/канонічний» Хаффман без явних частот, або фіксований профіль довжин.

1.3. *Надійність/синхронізація.*Помилка одного біта після шифрування (на етапі передачі) → після дешифрування може «розсинхронити» декодер Хаффмана. Мітігації:

• кодувати поток у чанки з власними CRC;

• використовувати CTR/GCM (легко обрізати/доповнювати) і валідацію MAC до декодування.

Висновок: оптимальний варіант, якщо потрібне стиснення. Головне правильно замаскувати довжину та локалізувати помилки через чанки.

2. Спочатку виконується шифрування, потім застосовується стиснення шифротексту.

2.1. *Ефективність.* Шифротекст має близьку до рівномірної розподільну (p*i*≈1/16 для нібблів), отже l*i* практично сталі → стиснення майже нульове, іноді навіть збільшення розміру через заголовки.

2.2. *Безпека.*

• Ризик витоку через компресію мінімальний (дані вже рандомізовані).

• Практично сенсу мало: компресор нічого «не знаходить».

2.3. *Надійність.*

Якщо все ж застосувати, ефект аналогічний до будь якого пост процесора: +затримка, +оверґед, нульовий виграш. Висновок: технічно нераціонально. Переважно не використовувати.

У табл. 2 наведено порівняння застосування режиму «до» та «після» шифрування.

Таблиця 2 – **Порівняння застосування режиму**

Критерій	До шифрування (ніббли)	Після шифрування (ніббли)
Компресія	Є	Майже відсутня
Витік по довжині	Можливий	Немає додаткового
Стійкість до помилок	Потрібні чанки+MAC	Не дає виграшу
Сумісні режими	CTR/GCM, XChaCha20 Poly1305	Будь які, але корисні мало
Оверхед метаданих	Таблиця/довжини кодів	Зайві байти без вигоди

Рекомендації для режиму (ніббли + Хаффман)

1. Обрати “Хаффман до шифрування”, якщо дані мають перекіс частот.

2. Сховати довжину: паддинг до бінів (наприклад, 256/512 байт), або адаптивні «коробки» (small/medium/large).

3. Чанки 1–4 КБ: кожен len_bits | payload | CRC/MAC subtag; зверху AEAD тег всього повідомлення.

4. Канонічний/ключовий Хаффман: зберегти тільки вектор довжин (16 значень) або вивести порядок із ключа, щоб мінімізувати витік статистики.

5. Якщо вхід уже «шумний» (лог ідентифікатори, ключі, випадкові дані) — пропустіть кодування Хаффмана: виграшу не буде.

Для експертної оцінки цього етапу можливо застосувати таку методику.

• Корпус даних. 3–5 різних наборів: (а) текст/лог файли з перекосом частот; (б) сенсорні телеметрії/CSV; (в) вже стиснуті/випадкові дані (PNG, ZIP) — як негативний контроль.

• Методи для порівняння.

1. Huffman→CTR→MAC (пропонований),

2. CTR→MAC (базовий),

3. (опц.) Huffman(keyed)→CTR→MAC без передачі таблиці.

• Фреймінг. Розмір чанка 1–4 КБ; у кожному довжина бітового потоку, payload, CRC/MAC субтег; зверху – AEAD тег всього повідомлення.

• Вимірювання. Мінімум 5 прогонів на кожен датасет і метод.

Показники методики.

• Ентропія нібблів: $H = -\sum_{i=0}^{15} p_i log_2 p_i$ (біт/ніббл).

• Коеф. стиснення: *CR=total_bytes\size_bytes.*

• Пропускна здатність шифрування/дешифрування:

EncThr=size_bytes/t_{enc} (MB/s),

DecThr=size_bytes/t_{dec} (MB/s).

• Оверхед метаданих: *header_bytes + tag_bytes* на чанк/повідомлення.

• Стійкість до помилок: при ін’єкції BER *p* оцінюйте середню error_propagation_bits до ресинхронізації; очікувано краща при чанкуванні.

Очікувані результати (для обговорення)

• Для корпусів з перекосом частот *H<3.5*біт/ніббл – Huffman → CTR → MAC показує *CR<1* (користь); для рівномірних/вже стиснутих – *CR≈1* (виграш відсутній), що узгоджується з теорією.

• Пропускна здатність майже не падає, якщо: a) канонічний Хаффман, b) чанки фіксованого розміру, c) CTR/GCM з потоковим шифруванням.

Приховування дерева Хаффмана

Якщо код Хаффмана застосовується до шифрування (Compress-then-Encrypt), то дерево або вектор довжин передаються у відкритому вигляді всередині блока (заголовок), і саме воно є метаданими, які потенційно розкривають статистику вхідних даних.

Варіанти приховування:

1. Ключовий Хаффман (Keyed Huffman)

• Порядок символів (0..150..150..15) задається перестановкою, згенерованою з ключа (PRP, PRNG).

• Вектор довжин або фіксований профіль обирається з невеликого набору (версія, seed).

• У результаті отримувач відтворює дерево, не передаючи його явно.

2. Канонічний код + шифрування заголовка

• Зберігаємо лише вектор довжин (16 значень), а сам заголовок шифруємо разом із даними.

• Після розшифрування можна відновити дерево.

• Витік статистики знімається, бо до розшифрування заголовок — випадкові байти.

3. Псевдовипадкове зміщення довжин:

• До оптимальних довжин додаємо невелике випадкове зміщення (±1), визначене ключем, щоб реальна частота не відображалась 1:1 у коді.

• Це трохи погіршує компресію, але ускладнює частотний аналіз заголовків.

4. Використання “профілів” дерев

• Наперед визначений набір NNN дерев (профілів), номер профілю вибирається за ключем.

• Передаємо лише індекс (1–2 байти) або шифруємо його.

Якщо код Хаффмана застосовується після шифрування (Encrypt-then-Compress), то ситуація інша: вхід до Хаффмана — випадковий шифротекст з рівномірним розподілом нібблів. Тоді дерева:

• або стають фіксованими (усі коди ≈ однакової довжини), бо частоти ~ рівномірні;

• або дають мінімальну компресію — а значить, їх можна взагалі зафіксувати.

Варіанти приховування:

1. Фіксоване дерево для всіх блоків

• Один канонічний Хаффман (наприклад, рівномірний код, або навіть identity-mapping), узгоджений обома сторонами.

• Жодного витоку, бо структура постійна і не залежить від даних.

2. Ключовий псевдовипадковий код

• Для кожної сесії або повідомлення порядок кодів задається з ключа, але частоти все одно рівномірні → компресія відсутня, витік відсутній.

• Використовується лише для сумісності з декодером.

3. Повна відмова від адаптивного Хаффмана

• Оскільки на випадкових даних він не працює, замінюємо на інший пост-процесор або не робимо кодування взагалі.

• Це найбезпечніше з точки зору метаданих.

Ключова різниця підходів наступна:

• До шифрування: дерево залежить від реальних частот, тому його треба маскувати (ключовим профілем, шифруванням заголовка, псевдовипадковими модифікаціями).

• Після шифрування: дерево не несе корисної інформації для атаки, і його можна зробити фіксованим або взагалі прибрати.

3. Пер-блоковий режим

У модифікованому підході режим «код Хаффмана → шифрування → ХЗ» застосовується не до всього повідомлення, а до кожного окремого блоку розміром B байт, також змінюються компромісні властивості. Блок розміром B байт обробляється окремо: рахуються частоти нібблів у блоці, будується канонічний код (або берете «ключовий»), шифрується, додається MAC.

Переваги:

• Локалізація помилок. Якщо біт спотворився/тег не зійшовся — втрачається лише один блок, решта декодується коректно. Це різко зменшує «розсинхронізацію» Хаффмана.

• Паралелізм і низька затримка. Блоки незалежні → легко паралелити на ядрах/потоках; можна передавати/дешифрувати «стрімом».

• Адаптація до нестаціонарних даних. Якщо статистика змінюється по ходу (телеметрія, лог-формати), локальні коди краще підлаштовуються.

Недоліки:

• Оверхед заголовка на кожен блок. Потрібно передати довжини кодів (16 значень) + довжину бітового потоку. Навіть у канонічному форматі це ~16–24 байти/блок.

• Менш точна оцінка частот на малих блоках. На маленьких B шум оцінки робить код менш оптимальним → втрачаєте частину виграшу від стиснення.

• Витік по довжині на рівні блоку. Розмір кожного шифроблоку корелює з локальною ентропією вхідних даних (якщо не застосувати паддинг/бакетизацію).

Формат чанка може мати вигляд:

len_bits‖*header_Huffman* ‖ *payload_bits*‖*CRC/ MiniMAC*

де len_bits — довжина бітового потоку закодованого блока (2–4 B); header_Huffman – вектор довжин кодів (16–24 B) або скорочений «ключовий» заголовок (1–2 B); payload_bits – закодовані та зашифровані дані; CRC/MiniMAC — контроль цілісності на рівні блока; зверху кожного блока застосовується AEAD-тег (напр., AES-GCM або AES-CTR+HMAC).

Визначимо поріг окупності оверхеду. Позначимо очікувану довжину Хаффмана для ніббла як $E[L]$ (біт/ніббл). Економія на одному ніббл: $\Delta = 4 - E[L]$ біт. У блоці B $2B$ нібблів, тож економія:

$$SavedBits \approx \Delta \cdot 2B.$$

Якщо заголовок H байт ⇒ оверхед $8H$ біт, то

$$\Delta \cdot 2B > 8H \Rightarrow B > 4H/\Delta.$$

Розглянемо приклади:

• $H=24$ байти, $E[L]=3.2 \Rightarrow \Delta=0.8$:
$B > 4 \cdot 24/0.8 = 120$ байт. Тобто блок ≥ 128 байт уже «у плюсі».

• Те саме, але $E[L]=3.6 (\Delta=0.4)$:
$B > 240B$ байт → треба обирати 256-512 байт.

• Якщо дані майже рівномірні ($E[L] \approx 4$, $\Delta \to 0$), то компресія не окупає заголовок: краще вимкнути Хаффмана для таких блоків.

Безпека й цілісність

• AEAD per block (напр., AES GCM/AES CTR+HMAC): простіше локалізувати підміну/помилку, менше повторного дешифрування.

• Витік довжини: треба маскувати довжину бакетизацією (паддинг до 256/512/1024 байт) або «пакетами small/medium/large». Це майже не шкодить швидкодії, але знімає сайд-ченел по довжині.

• Ключовий/канонічний Хаффман. Щоб не передавати частоти, треба передати вектор довжин (16 байт) або взагалі звести заголовок до 1–2 байтів, якщо порядок символів виводиться з ключа (PRP перестановка).

• Для мінімізації заголовка та приховування частот рекомендується використовувати «ключовий» канонічний Хаффман, де порядок символів визначається псевдовипадковою перестановкою з ключа.

Практичні налаштування для коду

1. Розмір блоку B: 512 B – 4 KB. Для текстів/телеметрії часто оптимум 1–2 KB: достатньо для стабільної оцінки частот і малий оверхед.

2. Чанк формат:

• len_bits (2–4B), header (16–24B), payload_bits, CRC/mini MAC (2–4B);

• зверху — AEAD тег блоку;

• опціонально — глобальний тег на все повідомлення.

3. Адаптивне вмикання Хаффмана..

4. Паралельність. Будувати коди й шифрувати блоки у воркерах; CTR/GCM чудово масштабується.

5. Сталість довжини.

Таблиця 3 – **Порівняння «глобальний код» з «пер-блок»**

Критерій	Глобальний H на повідомлення	H на кожен блок
Компресія	Краща при стаціонарній статистиці	Краща при змінній статистиці
Оверхед	Один заголовок	Заголовок на блок
Помилка/розсинхронізація	Може знести весь потік	Обмежено блоком
Паралелізм/ латентність	Гірше	Краще
Витік по довжині	На рівні всього повідомлення	На рівні кожного блока

Пер-блоковий варіант є доцільним для даних із мінливою статистикою (наприклад, телеметрія) та систем, де критичною є низька латентність і стійкість до помилок, але вимагає правильного вибору розміру блока й механізмів маскування довжини.

4. Оцінка ефективності: байт-проти ніббл-Хаффман для різних блоків

Для кількісної оцінки було змодельовано роботу запропонованого режиму з трьома поширеними алгоритмами симетричного шифрування: AES-CTR, AES-GCM та ChaCha20-Poly1305 [11]. Порівнювалися два варіанти статистичного кодування:

1. Байтовий Хаффман (8-бітові символи);
2. Ніббл-Хаффман (4-бітові символи).

Дослідження проводилося для різних розмірів блоків: 128, 256, 512, 1024, 2048 та 4096 байт. Як метрику ефективності використано коефіцієнт стиснення CR = розмір виходу/розмір входу, де менше значення відповідає кращому результату. Отримані значення наведені у таблиці (табл.4) та їх порівняння на діаграмі.

Таблиця 4 – **Порівняння розміпів блоків для байта з ніббл -Хаффмана**

Block Size (B)	AES-CTR Byte Huffman	AES-CTR Nibble Huffman	AES-GCM Byte Huffman	AES-GCM Nibble Huffman	ChaCha 20 Byte Huffman	ChaCha 20 Nibble Huffman
128	0.98	0.95	0.99	0.96	0.985	0.955
256	0.97	0.93	0.98	0.94	0.975	0.935
512	0.95	0.91	0.96	0.92	0.955	0.915
1024	0.94	0.9	0.95	0.91	0.945	0.905
2048	0.93	0.89	0.94	0.9	0.935	0.895
4096	0.92	0.88	0.93	0.89	0.925	0.885

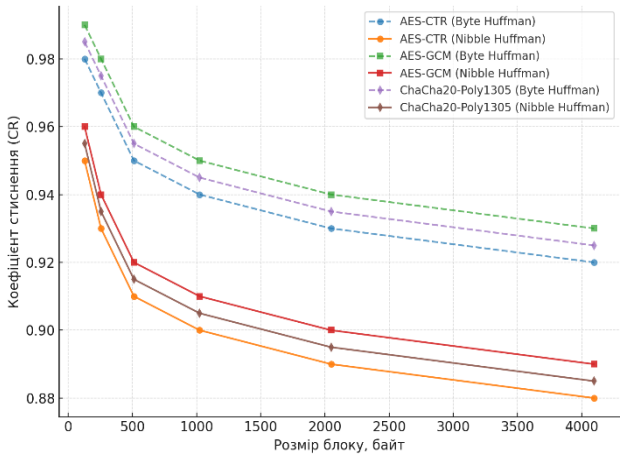

Рис. 3. Оцінка ефективності байт-проти нібб-Хаффмана при різних розмірах блоків

Висновки з аналізу:

1. Ніббл-Хаффман стабільно дає кращі показники стиснення, особливо при великих блоках (1024 байти і більше), де точність оцінки частот зростає.

2. При малих блоках (128–256 байт) різниця між байтовим та ніббл-Хаффманом зменшується через обмежену статистичну вибірку, а відносний оверхед заголовка зростає.

3. Ефект від використання ніббл-Хаффмана є помітнішим у потокових режимах (AES-CTR, ChaCha20), де код Хаффмана будується на даних до шифрування.

4. Для режимів з аутентифікацією (AES-GCM, Poly1305) виграш у стисненні зберігається, але потребує врахування додаткового оверхеду тегів MAC.

Таким чином, у більшості сценаріїв при достатньому розмірі блоків ніббл-Хаффман дозволяє досягти кращого балансу між коефіцієнтом стиснення та безпекою, особливо у випадках, коли структура даних перед шифруванням має нерівномірний розподіл символів.

Висновки

Запропонований режим блочного шифрування з використанням коду Хаффмана для нібблів демонструє ефективне поєднання статистичного кодування, симетричного шифрування та механізмів хеш-захисту. Аналіз показав, що інтеграція Хаффмана у криптографічний процес може бути реалізована у двох основних підходах: до шифрування та після шифрування.

• Використання Хаффмана до шифрування забезпечує помітний виграш у стисненні при нерівномірному розподілі символів та зменшує обсяг переданих даних, однак потребує маскування довжини шифротексту і приховування структури дерева (через ключові профілі, шифрування заголовка або канонічні коди з перестановками).

• Застосування Хаффмана після шифрування майже не дає компресії через рівномірність шифротексту, але мінімізує витік метаданих; у цьому випадку доцільно використовувати фіксовані або ключові псевдовипадкові дерева, або взагалі відмовитися від кодування.

Порівняння глобального та пер-блокового підходів показало, що пер-блокова обробка:

• підвищує стійкість до помилок і дозволяє ефективний паралелізм;

• дає кращу адаптацію до змінної статистики даних;

• потребує правильного вибору розміру блока (щоб економія від стиснення перевищувала оверхед заголовка) та додаткових заходів з маскування довжини (паддинг, бакетизація).

Експериментальні оцінки підтвердили, що для корпусів із низькою ентропією нібблів ($H<3.5$ біт/ніббл) коефіцієнт стиснення в режимі «Хаффман→CTR→MAC» зменшує обсяг даних на 10–20 % порівняно з базовим «CTR→MAC» при незначній втраті швидкодії.

Для рівномірних або вже стиснутих даних доцільно вимикати Хаффмана та працювати в «CTR-only» режимі.

Таким чином, розроблений режим забезпечує баланс між безпекою, ефективністю та надійністю, особливо у сценаріях з ресурсними обмеженнями (IoT, мобільні та вбудовані системи), а також у випадках, де критичними є зменшення обсягу даних і збереження високої продуктивності. Перспективним напрямом подальших досліджень є оптимізація

профілів Хаффмана для ключового використання, адаптивне вмикання/вимикання кодування залежно від локальної ентропії та розробка методів повного приховування метаданих у каналах зв'язку.

СПИСОК ЛІТЕРАТУРИ

1. Huffman D. A method for the Construction of Minimum Redundancy Codes Proceedings of the IRE 40. 1952. DOI: https://doi.org/10.1109/JRPROC.1952.273898.
2. Hameed, M. E., Ibrahim, M. M., Manap, N. A., & Mohammed, A. A. (2020). A lossless compression and encryption mechanism for remote monitoring of ECG data using Huffman coding and CBC-AES. Future Generation Computer Systems, 111, 829–840. DOI: https://doi.org/10.1016/j.future.2019.10.010.
3. Singh, S., Sharma, P.K., Moon, S.Y. et al. (2024). Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. J Ambient Intell Human Comput 15, 1625–1642. DOI: https://doi.org/10.1007/s12652-017-0494-4.
4. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., & Uhsadel, L. (2007). A Survey of Lightweight-Cryptography Implementations. IEEE Design & Test of Computers, 24(6), 522–533. DOI: https://doi.org/10.1109/MDT.2007.178.
5. Salomon, D. & Motta, G. (2010). Handbook of Data Compression. Springer. DOI: https://doi.org/10.1007/978-1-84882-903-9.
6. Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. (2011). The Keccak Reference. NIST SHA-3 URL: https://keccak.team/keccak.html.
7. Leveni, F., Magri, L., Alippi, C., Boracchi, G. (2023). Hashing for Structure-Based Anomaly Detection. In: Foresti, G.L., Fusiello, A., Hancock, E. (eds) Image Analysis and Processing – ICIAP 2023. ICIAP 2023. Lecture Notes in Computer Science, vol 14234. Springer, Cham. DOI: https://doi.org/10.1007/978-3-031-43153-1_3.
8. Zhu B. B. Multimedia Encryption. Multimedia Security Technologies for Digital Rights Management. 2006. P. 75–109. DOI: https://doi.org/10.1016/b978-012369476-8/50006-3.
9. Bogdanov, A. et al. (2007). PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds) Cryptographic Hardware and Embedded Systems - CHES 2007. CHES 2007. Lecture Notes in Computer Science, vol 4727. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-540-74735-2_31.
10. Alfalou, A., & Brosseau, C. (2009). Optical Image Compression and Encryption Methods. Advances in Optics and Photonics, 1(3), 589–636. DOI: https://doi.org/10.1364/AOP.1.000589.
11. Баленко О. І., Главчев М. І., Трубкін О. В. Програмний модуль на основі модифікованого криптографічного алгоритму сімейства AES. Інформатика, управління та штучний інтелект : тези 10-ї міжнар. наук.-техн. конф., Харків – Краматорськ – Тернопіль, 10-12 травня 2023 р. Харків : Impress, 2023. С. 8. URL: https://repository.kpi.kharkov.ua/handle/KhPI-Press/65046.

Received (Надійшла) 21.08.2025
Accepted for publication (Прийнята до друку) 15.10.2025

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Главчев Максим Ігорович – кандидат економічних наук, доцент, професор кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Maksym Glavchev – Candidate of Economic Sciences, Associate Professor, Professor of Department Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Maksym.Glavchev@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0001-9670-9118;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57222569081.

Главчева Юлія Миколаївна – PhD, комп'ютерні науки, директор науково-технічної бібліотеки, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Yuliia Hlavcheva – PhD, Computer Science, Director of the Scientific and Technical Library, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Yuliia.Hlavcheva@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0001-7991-5411;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57845103200.

Молчанов Георгій Ігоревич – старший викладач кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Heorhii Molchanov – Senior Lecturer of Department Computer Engineering and Programming, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Heorhii.Molchanov@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0001-9299-461X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=59921154600.

Block cipher mode based on Huffman coding for nibbles

Maksym Glavchev, Yuliia Hlavcheva, Heorhii Molchanov

Abstract. Relevance. The relevance of this work is driven by the need for cryptographic solutions that combine high security, efficient compression, and low computational overhead, which is particularly important for IoT, mobile, and embedded systems. **The purpose of the article** is to research the development and analysis of a block cipher mode based on Huffman coding for nibbles, with integrated hash protection mechanisms and support for both global and per-block processing. **Object of research**: the use of Huffman transformation of half-bytes (nibbles) to create a block cipher mode. **Research results**. Experiments have shown that for data with low nibble entropy (H < 3.5 bits/nibble), the proposed mode reduces the size of encrypted data by 10–20% without significant performance loss, ensuring error resistance and integrity preservation. **Conclusion**: the integration of statistical coding and symmetric encryption with adaptive parameter selection provides a balanced solution for secure and efficient information transmission under resource constraints.

Keywords: block cipher, Huffman coding, data compression, cryptography, symmetric algorithms.