# В**ступ**

В теперішній час одною з найважливіших проблем, пов’язаних із захистом конфіденційності інформації, що передається в інформаційно-телекомунікаційних системах, є потреба в генераторах псевдовипадкових послідовностей, що відповідають високим вимогам до рівномірності розподілу ймовірностей чисел, що формуються ними. Генератор ,слабкий з криптографічної точки зору, може значно ослабити захищеність інформаційної системи. Сьогодні відомо декілька сотень шифрів, які можуть ефективно використовуватися для захисту інформації. Незважаючи на це багато спеціалістів і організацій неперервно працюють над вдосконаленням відомих шифрів і пошуком нових. Президент України недавно заявив : Кіберпростір перетворився на ще одне поле протистояння і боротьби за незалежність держави. І це застереження без перебільшень дуже важливе. Слід нагадати, що група CERT-UA при Держспецзв'язку 2014 року зафіксувала 216 кібернападів ззовні (більше половини з них - на державні установи). У 2015 році число атак збільшилося у півтора разу. Найнебезпечніша атака відбулась 23 грудня, коли було завдано масованого вдару по диспетчеру компанії «Укренерго» і 6 енергокомпаніям, що призвело до відключення енергії у 103 населених пунктах на Заході країни. Американські експерти встановили, що ця масована кібердиверсія була здійснена з боку РФ . При проектуванні сучасної мікроелектронної техніки неможливо обходитись без криптографічних засобів захисту інформації. Це стосується мобільних телефонів, мобільних комп’ютерів, смартфонів та багато іншої мікроелектронної техніки, яка може використовуватися для передачі конфіденційної інформації. Так в 2016 році американська фірма WhatsApp вдосконалила і впровадила нову систему шифрування інформації(дзвінки,повідомлення,медіафайли,що передаються), яка за переконаннями авторів має високу стійкість і надійність. Тому очевидно, що дослідження і вдосконалення криптографічних інструментів захисту інформації є актуальне. Найбільш складна проблема для звичайного користувача – зрозуміти комплексну ефективність шифру, яка формується з його стійкості і простоти використання, продуктивності роботи при захисті інформації. Класична структура роботи зломщика шифрів – дослідження повторень в шифрованому тексті і дослідження статистичних характеристик частоти використання символів тексту.

В основу досліджень взяті блокові шифри, які базуються на матричних перетвореннях – множення квадратної матриці ключа на матрицю стовпчик відкритого тексту [1, 2]. Такі шифри довгий час вважались неперспективними, ручними і низькопродуктивними. З широким поширенням комп’ютерної техніки шифри на основі матричних перетворень отримали свою нішу і перспективу використання. Блокові шифри характеризуються (наприклад система шифрування мережа Фейстеля) такими параметрами: наявність властивостей перестановок, властивостей підстановок і статистичними характеристиками, які в деякій мірі наближаються до рівноймовірних.

# Основна частина

#### Хеш-функції

Хешування - перетворення вхідного масиву даних довільної довжини в вихідний бітову рядок фіксованої довжини, за певним алгоритмом. Таке перетворення також називається хеш-функція ( функція згортки), а його результат - хеш, хеш-код, хеш-сума ( зведення повідомлення).

Застосування хешування:

- Побудова асоціативних масивів;

- Пошук дублікатів в серіях наборів даних;

- Побудова досить унікальних ідентифікаторів для наборів даних;

- Контрольне підсумовування даних - для подальшого виявлення в них помилок (випадкових або навмисних), що виникають при зберіганні і (або) передачі сигналу;

- Зберігання паролів в системах захисту (в хешованому вигляді, - для відновлення з якого потрібна обробка функцією, яка є «зворотною» по відношенню до використаної хеш-функції);

- При виробленні електронного підпису (на практиці, - часто, - підписується не саме повідомлення, а його хеш-образ).

У загальному випадку (принцип Діріхле) - немає однозначної відповідності між вихідними даними і хеш-кодом, тому що значення хеш-функцій менш різноманітні, ніж значення вхідного масиву: різний зміст може дати однаковий хеш-код - що може привести до так званої колізії. Імовірність виникнення колізій грає важливу роль в оцінці якості хеш-функцій.

Як відомо, криптографічні хеш-суми відрізняються від звичайних хеш-сум тим, що крім основних властивостей, що вимагаються від будь-якої хеш-функції:

- здатності перетворення вхідного значення (зазвичай тексту) довільної довжини в вихідне значення фіксованої довжини,

- статистичної рівномірності випадаючих вихідних значень,

- хорошим «розкидом» (розбіжністю приблизно в половині біт) вихідних значень навіть для текстів, різниця між якими дуже незначна (можливо тільки в 1 біті);

до криптографічних хеш-алгоритмів пред'являються додаткові вимоги:

- розрядність вихідних значень повинна знаходитися далеко за межами можливостей повного перебору на сучасній техніці як за швидкістю обробки, так і за обсягом зберігання (на практиці це - розрядність в 128, 160, 256 і більше біт);

- не повинно існувати способу (істотно більш ефективного, ніж повний перебір вхідних значень) обчислити будь-яку пару вхідних текстів, що дають на виході однакове хеш-значення (неважливо яке) - успішна атака на цю вимогу називається «колізією» хеш-функції;

- не повинно існувати способу (істотно більш ефективного, ніж повний перебір вхідних значень) за значенням хеш-функції підібрати будь-який вхідний текст, що дає на виході алгоритму це хеш-значення - успішна атака на цю вимогу називається «оборотністю» хеш-функції.

Три перерахованих вище вимоги до крипостійкості хеш-функцій перетворюють їх за фактом в ідентифікатори фіксованої довжини для текстів, файлів і взагалі довільних блоків інформації. При цьому ці ідентифікатори:

- унікальні на всьому просторі сучасного інформаційного суспільства;

- незворотні, тобто не розкривають абсолютно нічого про зміст вихідного документа.

Це дозволяє використовувати хеш-функції в задачах:

- Перевірки цілісності файлів, архівів, збірок тощо (Досить довіреністю передати одержувачу хеш-суму файлу і тоді будь-які несанкціоновані зміни в файлі негайно змінять його контрольну хеш-суму);

- Зберігання паролів у незворотному вигляді (в сховище поміщаються не самі паролі, а хеш-суми від рядка, наприклад, виду «константа (соль)» + «пароль», що дозволяє при розкритті хешу зберегти власне словникову парольну фразу в секреті);

- Електронно-цифрового підпису документів (власне ЕЦП встановлюється завжди не на сам файл, що по-перше, було б дуже повільно, і по-друге, несло б деякі проблеми з безпекою самого ключа підпису, а на його хеш-суму, забезпечуючи такий ж рівень захисту від модифікацій).

На сьогоднішній день переважну частку застосувань хеш-функцій «беруть на себе» алгоритми MD5, SHA-1, SHA-256. Звичайно, існує і безліч інших менш відомих, або поширених тільки у вузьких співтовариствах алгоритмів (наприклад, RIPEMD, TIGER, PANAMA і ін.)

У криптографії сіль – це випадкова послідовність, що використовується в якості додаткових вхідних даних для хешування. Основна функція солі –захист від атак проти списку вихідних паролів за допомогою райдужних таблиць чи одного проходу повного перебору.

Нова сіль генерується випадковим чином для кожного пароля. У загальному випадку, сіль і пароль з’єднуються та обробляються за допомогою криптографічної хеш-функції ,і результуюча вихідна послідовність(але не оригінальний пароль) зберігається разом із сіллю в базі даних. Криптографічні солі широко використовуються в багатьох сучасних комп’ютерних системах від системи облікових даних Unix до інтернет-безпеки.

Райдужні таблиці представляють собою застосування більш раннього і простішого алгоритму Мартіна Хеллмана. Райдужна таблиця – це попередньо обчислена таблиця для реверсу криптографічної хеш-функції ,як правило ,для злому хеш-паролів. Таблиці здебільшого використовують для відновлення оригінального паролю певної довжини, що складається з обмеженого набору символів. Це практичний приклад time-memory tradeoff - розумного компромісу між часом пошуку по таблиці і займаною пам'яттю (використовується менше часу для комп’ютерної обробки інформації і більше пам’яті, ніж при brute-force атаці, але більше часу обробки і менше пам’яті, ніж при використанні таблиць пошуку).Використання ключа ,виведеного функцією з використанням солі робить такий вид атаки практично нездійсненним.

У криптографії, лавинний ефект відноситься до бажаної властивості криптографічних алгоритмів, зазвичай блокових шифрів та криптографічних хеш-функцій. Лавинний ефект очевидний у випадку, коли трохи змінена вхідна послідовність (наприклад, перевернутий один біт) спричиняє суттєві зміни вихідного сигналу (наприклад, половина вихідних біт обернулися). У випадку високоякісних блокових шифрів, така невелика зміна в будь-якому ключі або відкритому тексту повинна викликати різку зміну шифротексту. Сам термін був вперше використаний Хорстом Фейстелем , хоча концепція бере початок принаймні у дифузії Шеннона.

Якщо блокові шифри або криптографічна хеш-функція не проявляє лавинний ефект в значній мірі, то він має погану рандомізацію, і, таким чином, криптоаналітик може робити припущення про вхідний текст, якщо буде мати тільки вихідний. Це може бути достатньо, щоб частково або повністю зламати алгоритм. Таким чином, лавинний ефект є бажаною умовою з точки зору конструктора криптографічного алгоритму або пристрою.

При побудові шифру або хеш-функції значний прояв лавинного ефекту є однією з цілей первинного проектування. Це дозволяє невеликим змінам швидко поширюватися через ітерації алгоритму, таким чином, що кожен біт на виході повинен залежати від кожного біта входу до зупинки алгоритму

Криптографічна хеш-функція - будь-яка хеш-функція, яка є крипостійкою, тобто задовольняє ряд вимог, специфічних для криптографічних додатків.

Більшість криптографічних хеш-функцій приймають на виході рядок будь-якої довжини ,а на виході дають хеш-значення фіксованої довжини. Криптографічна хеш-функція повинна бути здатною витримувати всі відомі типи криптоаналітичних атак. Як мінімум вона мусить мати наступні властивості:

* Стійкість до відновлення прообразу(Для даного хеш-значення h має бути складно знайти повідомлення m,таке ,що h=hash(m).Це поняття пов’язане з поняттям односторонньої функції. Функції, які не володіють цією властивістю можуть вразливі до атак відновлення прообразу. Слід зауважити, що на даний момент немає використовуваних хеш-функцій з доведеною односпрямованістю. )
* Стійкість до відновлення другого прообразу(Для даного вхідного значення m1 має бути неможливо обчислювально знайти відмінне значення m2 таке, щоб hash(m1)==hash(m2).Функції, що не володіють цією рисою є вразливими до атаки відновлення другого прообразу )
* Стійкість до колізій(Повинно бути неможливо обчислювально знайти два відмінних повідомлення m1 і m2 таких, щоб hash(m1)=hash(m2).Така пара називається криптографічною хеш-колізією. Ця властивість іноді називають сильна колізійна стійкість. Вона вимагає, що хеш-значення має бути хоча б у 2 рази довше, ніж вимагає стійкість до повного перебору ; в іншому випадку колізії можуть бути знайдені за допомогою атаки «днів народження». Атака «днів народження» - вид криптографічної атаки, що використовує парадокс днів народження для злому шифрів чи пошуку колізій хеш-функцій. Зазвичай саме знаходження способу побудови колізій криптоаналітиками служить першим сигналом старіння алгоритму і необхідності його швидкої заміни.)

Дані вимоги не є незалежними:

- Оборотна функція нестійка до відновлення другого прообразу і колізій.

- Функція, нестійка до відновлення другого прообразу, нестійка до колізій; зворотні твердження невірні.

Слід зазначити, що досі ще не доведене існування незворотних хеш-функцій, для яких обчислення будь-якого прообразу заданого значення хеш-функції теоретично неможливо. Зазвичай знаходження зворотного значення є лише обчислювально складним завданням.

Наявність цих властивостей означає, що зловмисний противник не зможе замінити чи модифікувати вхідні дані без зміни хеш-значення. Тому, якщо два повідомлення мають однаковий хеш-код , то з високим ступенем вірогідності можна стверджувати, що вони ідентичні.

Функції, що відповідають цим критеріям все ж можуть мати інші небажані властивості. На теперішній час популярні хеш-функції можуть бути вразливими до атак довжини-розширення(коли дано значення hash(m) та довжина len(m),вибираючи підходящі m` зловмисник може обчислити hash(m || m`),де || означає конкатенацію ).Ця слабкість може бути використана для злому наївних схем автентифікації на основі хеш-функцій.

В ідеалі можна побажати ще сильніших умов. Для противника має бути неможливо знайти два повідомлення з істотно схожими значеннями хеш-коду; або вивести яку-небудь корисну інформацію про дані, з огляду лише на їх хеш-значення. Тому криптографічна хеш-функція повинна поводитися ,наскільки це можливо, як рандомна функція, зберігаючи при цьому детермінованість та ефективність обчислення.

Алгоритми контрольних сум, такі як CRC32 та інші циклічні перевірки надмірності призначенні для задоволення набагато слабших вимог і ,як правило, непридатні в якості криптографічних хеш-функцій. Наприклад CRC був використаний для перевірки цілісності повідомлень в стандарті шифрування WEP,але легко був відкритий тип атак, що використовував лінійність контрольних сум проти цього стандарту.

Буде розроблено такі елементи системи захисту інформації :система автентифікації та система для формування/перевірки електронного цифрового підпису. Вони будуть базуватися на самостійно розроблених на основі блокових шифрів криптографічної функції та шифру.

**Автентифікація** — процедура встановлення належності користувачеві інформації в системі пред'явленого ним [ідентифікатора](https://uk.wikipedia.org/wiki/%D0%86%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%82%D0%BE%D1%80).

***Принцип роботи системи автентифікації***

В більшості випадків паролі не зберігаються на цільових об’єктах, а зберігається лише їх хеш-код. Зберігати парольні фрази недоцільно, тому що у випадку несанкціонованого доступу до файлу з фразами, зловмисник дістане інформацію про всі паролі та зможе ними скористатися , а при збереженні хеш-коду він здобуде лише цей код, який не може бути перетворений у вхідні дані, в даному випадку в парольну фразу. Під час процедур автентифікації обчислюється хеш-значення введеної парольної фрази і порівнюється із збереженим.

Прикладом в даному випадку служать ОС Linux та Windows.В них зберігаються лише хеш-значення парольних фраз із облікових записів користувачів.

Така система має на увазі передачу повідомлення по захищеному каналу, тобто по каналу, із якого криптоаналітик не може перехопити повідомлення чи послати своє. В іншому випадку він зможе перехопити пароль та використовувати його для подальшої нелегальної автентифікації. Захищатися від подібних атак можна за допомогою методу «виклик-відповідь». Цей метод я буду використовувати при розробці системи автентифікації.

***Приклад роботи системи автентифікації***

Нехай деякий користувач з іменем NAME проводить автентифікацію по парольній фразі,PASS .У базі даних зберігається значення розробленої криптографічної хеш-функції HASH(PASS,S),де S - сіль(псевдовипадкова послідовність визначеної довжини).Клієнт посилає запит(NAME,R),де R – псевдовипадкова послідовність певної довжини. У відповідь з бази даних отримується значення S.Користувач вираховує значення хеш-функції HASH(R,HASH(PASS,S)) і посилає на обробку. Обробник також обчислює значення HASH(R,HASH(PASS,S)) та порівнює його з отриманим. Якщо значення співпадають – автентифікація вірна.

При такому способі пароль не зберігається відкрито в базі даних і ,навіть перехопивши всі повідомлення між користувачем та обробником, криптоаналітик не в змозі відновити значення паролю ,а хеш-значення, що передається, кожен раз інше.

**Електронний цифровий підпис (ЕЦП)** — вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача. Електронний цифровий підпис накладається за допомогою особистого(секретного) ключа та перевіряється за допомогою відкритого ключа.

Варто зауважити один дуже важливий аспект: електронний цифровий підпис не існує в готовому вигляді ніколи. Він не існує у вигляді файлу чи чого-небудь ще. Електронний цифровий підпис насправді з’являється, створюється тільки в той момент часу, коли документ підписується секретним закритим ключем. Тобто, документ не підписується електронним цифровим підписом. Це в корені неправильний вираз. Документ підписується тільки закритим ключом, а не електронним цифровим підписом тільки після підписання секретним ключем ,тільки в цей момент часу створюється електронний цифровий підпис. Загальний механізм створення такий :береться будь-який документ, будь-якої довжини, розраховується його контрольна сума(хеш значення),і це значення шифрується закритим ключом. У підсумку, отриманий шифрований рядок і є електронним цифровий підписом.

***Принцип роботи системи створення/перевірки електронного цифрового підпису***

*Створення ЕЦП*

Для документа за допомогою спеціального комп’ютерного забезпечення вираховується ,так званий, дайджест повідомлення. В даному випадку це хеш-значення(може бути контрольною сумою тощо).

Дайджест характеризується такими властивостями:

1.Фіксована,чітко визначена довжина, що не залежить від розміру повідомлення

2.Відбиток повідомлення(дайджест) є унікальним для кожного повідомлення

3.По дайджесту неможливо відновити повідомлення

Таким чином, якщо документ був модифікований, то зміниться і його відбиток, що відобразиться при перевірці електронного цифрового підпису.

Далі за допомогою розробленого алгоритму шифрування відбиток документу кодується закритим ключем. Отриманий електронний цифровий підпис передається окремо разом із документом.

Розшифрувати електронний цифровий підпис і отримати початковий дайджест повідомлення можна тільки використовуючи відкритий ключ автора.

Тобто, обчислення дайджеста документа захищає його від модифікації сторонніми особами після підписання, а шифрування секретним особистим ключем підтверджує авторство документу.

*Перевірка справжності підписаного документа*

Перевірка електронного цифрового підпису документа здійснюється в три етапи:

1.Перший етап. Адресат, використовуючи необхідне програмне забезпечення та відкритий ключ автора, розшифровує підписує відбиток і отримує дайджест початкового документа.

2.Другий етап. За допомогою відповідної програми та спеціальної математичної функції з отриманого документа обчислюється його дайджест.

3.Третій етап. При перевірці електронного цифрового підпису порівнюються дайджести одержаного та початкового документів. В результаті перевірки ми можемо зробити висновок про справжність документа.

Розробка криптографічної функції

Шифрувальний алгоритм вибраний на основі шифру Хілла.

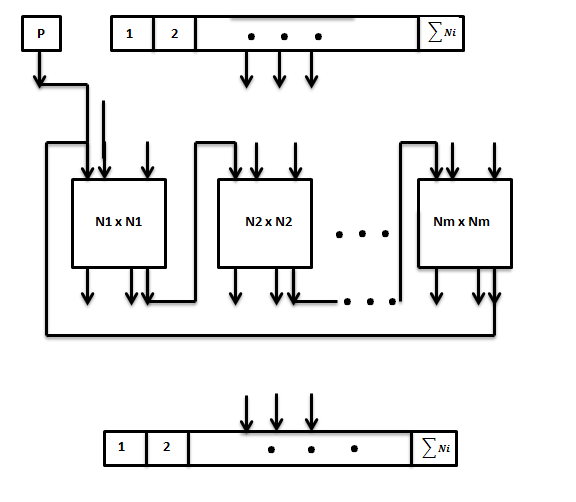


Рис.1.Схема функціонування алгоритму

Блок Ni x Ni - одна з матриць-ключів.P – символ ,що подається на вхід незалежно від даних. Вхідний блок повідомлення доповнюється до потрібної довжини ∑Ni-m (якщо потрібно).Далі він дробиться на підблоки довжиною Ni -1 і подається на обробку. Відбувається множення рядка(перший символ – так званий ,вхідний, решта – відповідний підблок) на відповідну матрицю-ключ з масиву матриць. На виході отримуємо рядок(останній символ – вихідний, решта результуючий підблок).Як можна побачити із рисунку ….. ,окремо ми подаємо лише перший вхідний символ, а далі вхідний символ підблоку – це вихідний символ попереднього підблоку.З вихідних рядків беремо результуючі підблоки та даємо їх на вихід алгоритму(або на наступне коло обробки).Кількість циклів обробки попередньо вказуємо.

Оскільки криптографічна хеш-функція повинна продукувати значення сталого розміру, необхідно застосовувати стиснення. Стиснення відбувається за дещо спрощеною схемою Міагучі-Пренеля:

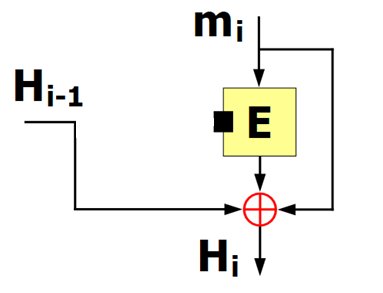


Рис.1.Спрощена схема Міагучі-Пренеля

Одностороння функція стиснення Міагучі-Пренеля (The Miyaguchi-Preneel single-block-length one-way compression function) є розширеним варіантом аналогічної функції Matyas-Meyer-Oseas. Це була функція, незалежно запропонована Бартом Пренелем і Сьодзо Міагучі.

Функція подає кожен блок вихідного повідомлення (mi) в якості тексту, який буде зашифрований. Далі виконується операція XOR зашифрованого тексту з тим же блоком вхідного повідомлення (mi), а потім також операція XOR з попереднім значенням хеш-функції (Hi-1) для отримання наступного значення хешу (Hi).

Попереднє значення хеш-функції (Hi-1) подають як ключ для блочного шифру. У першому раунді, коли немає попереднього значення хеш, він використовує постійне заздалегідь обумовлений початкове значення (H0).

Якщо блоковий шифр має різні розміри блоку і ключа, тоді значення хеш-функції (Hi-1) буде мати неправильний розмір для використання в якості ключа. Хеш-функція подається на функцію g () для перетворення, щоб відповідати як ключ для шифрування.

Дана схема не призначення для ключів блоків. Слідуючи їй було б потрібно кожного проходу використовувати функцію, що перетворювала б ключ-рядок в ключ-множину матриць. Це спричиняє зайві труднощі та витрати ,тому в моєму варіанті ключ залишається незмінним та є множиною матриць різних розмірів.

Математично це може бути представлено так:

Перевірка якості розробленої криптографічної хеш-функції

1.Стійкість для відновлення образів.

Довжина одного блоку хеш-коду – 32 , символів, доступних для формування послідовності – 64.Тобто,можливих значень – 6432 ≈ 6.3 \*1057.Припустимо,що можна перебирати значення зі швидкістю 1012 штук за секунду. Тоді для повного перебору даної кількості значень буде необхідно 6.3\*1045секунд = 175\*1040 годин = 20\*1037 років. Оскільки алгоритм новий, досі ще не існує ефективніших способів, ніж повний перебір для знаходження потрібного значення. Тобто хеш-функція досить добре протистоїть атакам відновлення образів.

2.Стійкість до колізій.

Найпоширеніша атака проти цієї властивості – атака днів народження. Одним із наслідків парадоксу днів народження є те, що для n-бітового блокового шифру повторювані появи блоку шифротексту можуть очікуватися з вірогідністю близько 0,63 при наявності лише 2n/2 випадкових відкритих текстів, зашифрованих на одному ключі (незалежно від розміру ключа).Тобто, щоб з високою ймовірністю можна було знайти колізії, необхідно перебрати 2n/2 варіантів. У даному випадку n =192,тому потрібно перебрати 286 можливих комбінацій ≈7.7 \*1025. Припустимо, що можна перебирати значення зі швидкістю 1012 штук за секунду. Тоді для повного перебору даної кількості значень буде необхідно 7.7\*1013секунд = 213\*108 годин = 24\*105 років. Тобто хеш-функція досить добре протистоїть колізійним атакам.

3.Лавинний ефект.

Результати виконання хешування даних, що мінімально відрізняються одне від одного.

|  |  |  |
| --- | --- | --- |
| Вхідне значення | Хеш-значення | К-ть змінених бітів |
| MAHEROVSKYA | #<$\_P$.2G!XZ\_Q>YGCD=1[MY,PD0I3}U | 0 із 192 |
| MAHEROVSKYB | "|!Z<U1Z1N$ST H"05N]%-6J;$AE[]LS | 101 із 192 |
| MAHEROVSKYC | })N\*3)7Q=.LL94UG>IX,DRA>-T\_8O\*8Q | 85 із 192 |
| MAHEROVSKYD | {(.RN {HX95E!}1;U[1MUE ['\*>M$?FO | 103 із 192 |
| MAHEROVSKYE | ]\*9{)\_^?%)!\_V|}&}O[=5)&,0X)#UH2M | 68 із 192 |
| MAHEROVSKYF | [&)J10<\*NFZ([C+LI$\*](@ON1<&U)Q!K | 102 із 192 |
| MAHEROVSKYG | @^F5LD!]8T^"BMI22U!,H;8?2.$| Z,I | 92 із 192 |
| MAHEROVSKYH | 9%TB\*6F3D1N8XWV<=)JMYQC"3\_},\_2<G | 106 із 192 |
| MAHEROVSKYI | 8$1;'JL -"71{02QW T=9D,04'[A0[WE | 52 із 192 |
| MAHEROVSKYJ | 7#"<J]RR<\_B D@"7#\_;]>((R5B84D&^C | 108 із 192 |
| MAHEROVSKYK | 6"\_V^PXIUL.TZ&=?K07,L9QC635I6\_QA | 86 із 192 |
| MAHEROVSKYL | 5}L$;%;!"Z\*M"?JV4D$M,,@^7F2]JG{? | 92 із 192 |
| MAHEROVSKYM | 4{ZNHV3(K7PFFIW]?6+={PE487'Q]PK= | 72 із 192 |
| MAHEROVSKYN | 3]79$|9{5\*9?.S3DYJF]?C-V9J,&PY7> | 100 із 192 |
| MAHEROVSKYO | 2[\*F.."4ADD)$,# %]P,P\*|G@[ZY%1E| | 85 із 192 |
| MAHEROVSKYP | 1@D1F?\*..R;#H6\_$MPZM08S)[NW=V@1( | 98 із 192 |
| MAHEROVSKYQ | 09R\_"1+S(')9;#KI6%3=$.]8]"T0|^?& | 38 із 192 |
| MAHEROVSKYR | '8'ZZEBJR{R2^>X'AV{]COGZ{RQE.=.% | 100 із 192 |
| MAHEROVSKYS | -7{\*D7HA]+[.JE4( |),TB0K}^N8?F|# | 89 із 192 |
| MAHEROVSKYT | ;6+R{KN)HJFU'O$N&.BM4&>+"VKM1OV} | 100 із 192 |
| MAHEROVSKYU | ,5J{X{T}2X'N\*Y?4O?L=\*7U]#)H#EX%] | 66 із 192 |
| MAHEROVSKYV | .4XJBQZ5\_5<GL2L+81V]G };$ZEU70P@ | 102 із 192 |
| MAHEROVSKYW | 355[^',Y^T!1]YSCE',XNIO%+B|K9]8 | 76 із 192 |
| MAHEROVSKYX | Z2^BVW5T^B{|)(59,79M8A2!^;?,{%J6 | 108 із 192 |
| MAHEROVSKYY | Y1B;!<[KOPH$NA%A(K^=<^=#&!+AQ+64 | 54 із 192 |
| MAHEROVSKYZ | X0P<9,$B9;1@3K!XQ{\_]K6W1\*1|4^ED2 | 112 із 192 |
| MAHEROVSKY | W';VT!)|E[+3<UM}@QH,.Z#S(D\*IWN00 | 88 із 192 |
| MAHEROVSKY. | V-[$\_2\_"'<V,P-ZFE^RM]MKD)5%]<W\_- | 90 із 192 |
| MAHEROVSKY, | U;<N7FD6>H"V586,-W.=\_!4&|H"Q,' , | 70 із 192 |
| MAHEROVSKY; | T,H9R8J;VVJO+%^^|<5]O%?5<9]&!8) | 108 із 192 |
| MAHEROVSKY- | S.VF+LPU#33HR=AKS,",'5YW>L9Y2$UY | 92 із 192 |
| MAHEROVSKY' | R 315}VLL$\_A7GN1]!<M#Y%H+{6=F>$W | 105 із 192 |
| MAHEROVSKY0 | QZ$\_PR.C6!X<\_Q |G2D=BLM|=P308DOU | 17 із 192 |
| MAHEROVSKY1 | PY!Z<&1<BN$%T 7P0FN]S?69\_$0ELM[S | 100 із 192 |
| MAHEROVSKY2 | OXN\*3X7#,.L[94&6>8X,3$A ?T;8}VIQ | 88 із 192 |
| MAHEROVSKY3 | NW.RN>{7)954!}B\_UL1M&4 L!\* MR-5O | 102 із 192 |
| MAHEROVSKY4 | MV9{);^-S)!;V|OU}}[=FX&=AXX#&7CM | 73 із 192 |
| MAHEROVSKY5 | LU)J1A<V{FZW[C.[IR\*]WKO{B<UUX#'K | 107 із 192 |
| MAHEROVSKY6 | KTF5L3!MIT^PBM8C2&!,7\_8-C.R|><=I | 85 із 192 |
| MAHEROVSKY7 | JSTB\*GFD31NIXW\*Z=XJM|#CPD\_O,;CZG | 103 із 192 |
| MAHEROVSKY8 | IR1;'9L>?"7B{0C#W>T=J3,AE'LAAL(E | 51 із 192 |
| MAHEROVSKY9 | HQ"<JMR$Z\_B>D@PH#;;] W($FBI43UTC | 113 із 192 |
| MAHEROVSKY@ | GP\_V^"X8&L.^Z&,-KA7,[JQ2G3FIG;#A | 85 із 192 |
| MAHEROVSKY[ | FOL$;S;'PZ\*]"?9\*43$M==@THFC]96N? | 93 із 192 |
| MAHEROVSKY] | ENZNH\*3W@7P5FI(M?G+=N"EEI7!QM"@= | 69 із 192 |
| MAHEROVSKY{ | DM79$Y9NF\*9-.SD3Y9F]-2-\*JJ=&"|H> | 107 із 192 |
| MAHEROVSKY} | CL\*F.+"E0DDX$,Q>%MP,"V|6K[<YSB4| | 88 із 192 |
| MAHEROVSKY" | BKD1F-\*++R;QH6;RM"ZMAISXLN(=\*KB( | 105 із 192 |
| MAHEROVSKY# | AJR\_"B+%W')J;#@86S3=R+]IM"^0YT-& | 37 із 192 |
| MAHEROVSKY$ | !I'ZZ4B9${RC^>)!A\*{]2}G<NR#E+,+% | 99 із 192 |
| MAHEROVSKY% | ?H{\*DHH0M+[+JEEW Y),^10@O^{8-5Y# | 86 із 192 |
| MAHEROVSKY^ | \_G+R{@NX7JF&'OR{&+BMEU>.PV@MB}\*} | 103 із 192 |
| MAHEROVSKY& | =FJ{XNTOCX'{\*Y-EO-L=VHUMQ)7#4)S] | 69 із 192 |
| MAHEROVSKY\* | +EXJB#ZF;5<6L2[.8BV]6>}\_RZ4UHA"@ | 97 із 192 |
| MAHEROVSKY( | >D55[T'=|^T'1]|%C4',){I}S+1|@JM8 | 79 із 192 |
| MAHEROVSKY) | <C^BV(5^TB{Y)(FJ,H9MI02'T;-,NS96 | 99 із 192 |
| MAHEROVSKY| | |BB;!Z[@}PHRNAS0(@^=ZT=QU!.A#.G4 | 59 із 192 |
| MAHEROVSKY< | )AP<9=$1J;1K3K')QN\_]@GWBV1Y4T432 | 103 із 192 |
| MAHEROVSKY> | (!;VT')Y4[+D<U]O@#H,+<#%WDVI({A0 | 85 із 192 |
| MAHEROVSKY+ | \*?[$\_C\_P!<V=P-<5ETRMM]K3X5S]Z(;- | 89 із 192 |
| MAHEROVSKY= | &\_<N75DG H"\*58G=-(.=;'4UYHPQ=!>, | 69 із 192 |
| MAHEROVSKY\_ | ^=H9RIJ\_\*VJ}+%TT|Z5]}S?FZ9M&'IX | 105 із 192 |
| MAHEROVSKY? | %+VF+[P&Q337R=0@S=",!FY( LJYCR&Y | 87 із 192 |
| MAHEROVSKY! | $>315OV[[$\_07G{B]'<MQ|%7.{G=5 RW | 104 із 192 |

Табл.1.Результати виконання хешування

Як видно з наведеної вище таблиці, внаслідок зміни одного біта у вхідній послідовності були спричинені значні зміни у вихідній. Тобто дана хеш-функція явно демонструє лавинний ефект.

4.Рівномірність розподілу частот.



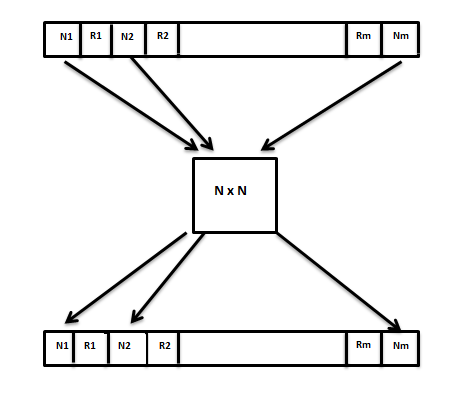
Рис.1.Частотний розподіл символів у тексті, шифрованому хеш-функцією.

Для даного аналізу було використано шифрувальний алгоритм хеш-функції без стиснення(щоб можна було оцінити частотні характеристики результатів)

Визначимо середнє інтегральне відхилення за допомогою формули 3.1.

Середнє інтегральне відхилення дорівнює 12.52 % (для порівняння – стандартний метод Хілла дає відхилення близько 30%).По рис.1. можна побачити, що частотний розподіл символів достатньо близький до рівномірного, водночас процес хешування не вимагає значних ресурсів для виконання.

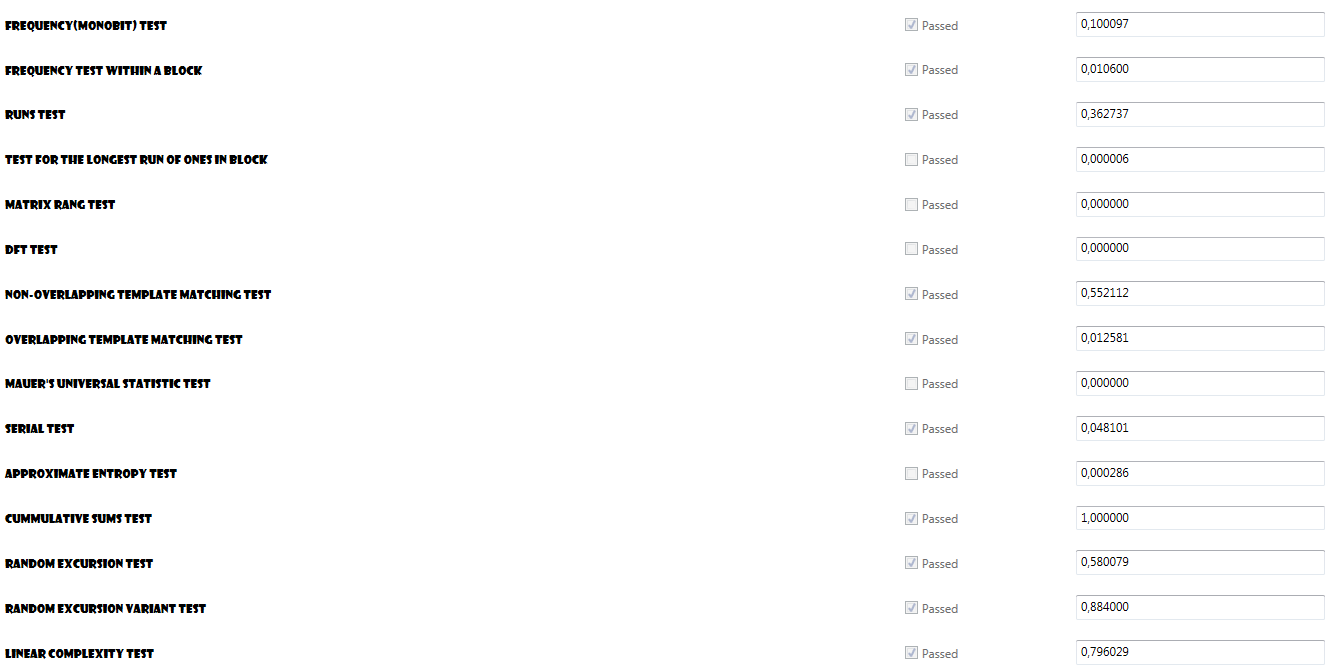
Розробка шифра



Рис…Схема функціонування алгоритму

Блок N x N – матриця-ключ.Ni – підблок повідомлення розміром N(подається на обробку), Ri – підблок повідомлення рандомної довжини(не подається на обробку) .

Блок Ni x Ni - одна з матриць-ключів.P – символ ,що подається на вхід незалежно від даних. Вхідний блок повідомлення доповнюється до потрібної довжини ∑Ni-m (якщо потрібно).Далі він дробиться на підблоки довжиною Ni -1 і подається на обробку. Відбувається множення рядка(перший символ – так званий ,вхідний, решта – відповідний підблок) на відповідну матрицю-ключ з масиву матриць. На виході отримуємо рядок(останній символ – вихідний, решта результуючий підблок).Як можна побачити із рисунку ….. ,окремо ми подаємо лише перший вхідний символ, а далі вхідний символ підблоку – це вихідний символ попереднього підблоку.З вихідних рядків беремо результуючі підблоки та даємо їх на вихід алгоритму(або на наступне коло обробки).Кількість циклів обробки попередньо вказуємо.



Рис….Результати NIST тестування.

# Список Літератури

# А. П. Алферов, Основы криптографии. Брюс Шнайер, Прикладная криптография.