**Лабораторна робота № 2**

**СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ІНФОРМАЦІЇ**

**Мета роботи:** ознайомитись з методами криптографічного забезпечення цілісності інформації, навчитись створювати програмні засоби для забезпечення цілісності інформації з використанням алгоритмів хешування.

**Теоретичні відомості**

Хеш функція – це функція, що відображає вхідне слово скінченної довжини у скінченному алфавіті в слово заданої, зазвичай фіксованої довжини. Таким чином, [функція хешування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187824&displayformat=dictionary) отримує на вхід повідомлення *M* довільної довжини, а на вихід видає [хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary) *H*(*M*) фіксованого розміру, який іноді називають профілем повідомлення. [Хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary) є функцією усіх бітів повідомлення і забезпечує можливість контролю помилок: зміна будь-якої кількості бітів повідомлення призводить до зміни [хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary)у.

Основні області використання хеш функцій – [аутентифікація](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187756&displayformat=dictionary) інформації та цифровий підпис. Практичне використання функцій хешування накладає на них ряд вимог, наведених нижче:

1.     Хеш-функція *H* повинна застосовуватися до блоку даних будь-якої довжини.

2.     Хеш-функція *H* створює вихід фіксованої довжини.

3.     *H*(*M*) відносно легко (за поліноміальний час) обчислюється для будь-якого значення *M*, а алгоритм обчислення повинен бути практичним з погляду як апаратної, так і програмної реалізації.

4.     Для будь-якого даного значення [хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary)у *h* обчислювально неможливо знайти *M* таке, що *H*(*M*)=*h*. Таку властивість іноді називають **односторонністю**.

5.     Для будь-якого даного блоку *x* обчислювально неможливо знайти *y*≠*x*, для якого *H*(*x*)=*H*(*y*). Таку властивість іноді називають **слабкою опірністю**[**колізія**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187782&displayformat=dictionary)**м**.

6.     Обчислювально неможливо знайти довільну пару різних значень *x* та *y*, для яких *H*(*x*)=*H*(*y*). Таку властивість іноді називають **сильною опірністю**[**колізія**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187782&displayformat=dictionary)**м**.

Перші три властивості описують вимоги, що забезпечують можливість практичного застосування функції хешування для аутентифікації повідомлень.

Четверту властивість визначає вимога односторонності хеш-функції: легко створити [хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary) за даним повідомленням, але неможливо відновити повідомлення за даним [хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary)ом.

П'ята властивість гарантує, що неможливо знайти інше повідомлення, значення хеш-функції якого збігалося б зі значенням хеш-функції даного повідомлення. Це запобігає підробці аутентифікатора при використанні зашифрованого [хеш-код](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187825&displayformat=dictionary)у.

Шоста властивість захищає проти класу атак, побудованих на парадоксі задачі про дні народження. Вона унеможливлює знаходження двох довільних різних повідомлень з однаковим хешем.

**Алгоритм хешування**[**MD5**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary)

Алгоритм [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary) обчислення профілю повідомлення (описаний в RFC 1321) був розроблений Роном Райвестом (Ron Rivest) з Масачусетського технологічного інституту (MIT) на початку 90-х років. До недавнього часу [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary) залишався найбільш поширеним із захищених алгоритмів хешування і використовувався як в найпопулярніших операційних системах Windows, Linux, BSD, так і в системах захисту інформації, таких як PGP. На даний час основне використання цього алгоритму – криптографічні контрольні суми, наприклад для перевірки цілісності отриманих з мережі Інтернет файлів.

Алгоритм одержує на вході повідомлення довільної довжини (довжина в бітах може дорівнювати нулю, вона не обов'язково повинна бути кратною восьми) й створює в якості виходу хеш повідомлення довжиною 128 біт. Алгоритм складається з наступних кроків:

***Крок 1: додавання доповнення***

Повідомлення доповнюється таким чином, щоб його довжина стала рівною 448 по модулю 512 (довжина 448 mod 512). Це означає, що довжина доданого повідомлення на 64 біта менша, ніж число, кратне 512. Додавання здійснюється завжди, навіть якщо повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 бітів, воно доповнюється 512 бітами до 960 бітів. Таким чином додається від 1 до 512 бітів.

Структура бітів доповнення наступна: перший біт дорівнює 1, а усі решта дорівнюють 0.

***Крок 2: додавання значення довжини***

64-бітне значення довжини вихідного (до доповнення) повідомлення в бітах приєднується до результату першого кроку. Якщо початкова довжина більша, ніж 264, то використовуються тільки молодші 64 біта. Таким чином, поле містить довжину вихідного повідомлення по модулю 264. Структура розширеного повідомлення схематично зображена на рис. 1.

Рис. 1. Структура розширеного повідомлення

У результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам. Це розширене повідомлення представляється як послідовність 512-бітних блоків *Y*0,*Y*1,…,*Y*L-1 (рис. 2), при цьому загальна довжина розширеного повідомлення дорівнює (*L*512) бітів. Таким чином, довжина отриманого розширеного повідомлення кратна довжині блоку з шістнадцяти 32-бітних слів.

Рис. 2. Загальна схема виконання алгоритму [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary)

***Крок 3: ініціалізація MD-буфера***

Для зберігання проміжних і кінцевих результатів функції хешування використовується 128-бітний буфер. Буфер може бути представлений як чотири 32-бітних регістри (A, B, C, D). Ці регістри ініціалізуються наступними 32-бітовими цілими значеннями (в шістнадцятковому представленні):

А = 0x67452301

В = 0xEFCDAB89

C = 0x98BADCFE

D = 0x10325476

Ці значення зберігаються у форматі прямого порядку запису байтів, коли найменш значущий байт слова знаходиться в позиції з молодшою адресою. Як 32-бітові рядки, значення ініціалізації (в шістнадцятковому запису) матимуть вигляд:

слово А: 01  23  45  67,

слово B: 89  AB  CD  EF,

слово C: FE  DC  BA  98,

слово D: 76  54  32  10.

***Крок 4: обробка повідомлення блоками по 512 бітів (16-слів)***

Основою алгоритму є функція стиснення – модуль, що складається із чотирьох циклічних обробок, позначений на рис. 2 як HMD5, а логіка роботи модуля показана на рис. 3. Чотири цикли мають схожу структуру, але кожний цикл використовує свою елементарну логічну функцію, позначену *fF, fG, fH, fI* відповідно.

Рис. 3. Обробка чергового 512-бітового блоку

Кожен цикл приймає в якості входу поточний 512-бітовий блок *Yq* і 128-бітове значення буфера ABCD, в якому міститься проміжне значення профілю, і змінює вміст цього буфера. Кожний цикл також використовує четверту частину 64-елементної таблиці *T*[1…64], складеної зі значень функції синуса. У таблиці *T* *i*-ий елемент, позначений *T*[*i*], має значення, що дорівнює цілій частині числа 232|sin(*i*)|, де *i* відповідає значенню кута в радіанах. Оскільки |sin(*i*)| лежить у діапазоні між 0 і 1, кожен елемент *T* є цілим числом, яке може бути представлено 32 бітами. Таблиця забезпечує "випадкову" множину 32-бітних значень, які повинні ліквідувати будь-яку регулярність у вхідних даних. Список значень *T* наведено в табл. 1.

Вихід четвертого циклу додається по модулю 232 до входу першого циклу (*MDq*), в результаті чого отримують *MDq+1*. Додавання виконується незалежно для кожного з чотирьох слів у буфері.

Таблиця 1. Значення *T*, створені на основі функції синуса

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***i*** | ***T*[*i*]** | ***i*** | ***T*[*i*]** | ***i*** | ***T*[*i*]** | ***i*** | ***T*[*i*]** |
| 1. | 0xd76aa478 | 17. | 0xf61e2562 | 33. | 0xfffa3942 | 49. | 0xf4292244 |
| 2. | 0xe8c7b756 | 18. | 0xc040b340 | 34. | 0x8771f681 | 50. | 0x432aff97 |
| 3. | 0x242070db | 19. | 0x265e5a51 | 35. | 0x6d9d6122 | 51. | 0xab9423a7 |
| 4. | 0xc1bdceee | 20. | 0xe9b6c7aa | 36. | 0xfde5380c | 52. | 0xfc93a039 |
| 5. | 0xf57c0faf | 21. | 0xd62f105d | 37. | 0xa4beea44 | 53. | 0x655b59c3 |
| 6. | 0x4787c62a | 22. | 0x2441453 | 38. | 0x4bdecfa9 | 54. | 0x8f0ccc92 |
| 7. | 0xa8304613 | 23. | 0xd8a1e681 | 39. | 0xf6bb4b60 | 55. | 0xffeff47d |
| 8. | 0xfd469501 | 24. | 0xe7d3fbc8 | 40. | 0xbebfbc70 | 56. | 0x85845dd1 |
| 9. | 0x698098d8 | 25. | 0x21e1cde6 | 41. | 0x289b7ec6 | 57. | 0x6fa87e4f |
| 10. | 0x8b44f7af | 26. | 0xc33707d6 | 42. | 0xeaa127fa | 58. | 0xfe2ce6e0 |
| 11. | 0xffff5bb1 | 27. | 0xf4d50d87 | 43. | 0xd4ef3085 | 59. | 0xa3014314 |
| 12. | 0x895cd7be | 28. | 0x455a14ed | 44. | 0x4881d05 | 60. | 0x4e0811a1 |
| 13. | 0x6b901122 | 29. | 0xa9e3e905 | 45. | 0xd9d4d039 | 61. | 0xf7537e82 |
| 14. | 0xfd987193 | 30. | 0xfcefa3f8 | 46. | 0xe6db99e5 | 62. | 0xbd3af235 |
| 15. | 0xa679438e | 31. | 0x676f02d9 | 47. | 0x1fa27cf8 | 63. | 0x2ad7d2bb |
| 16. | 0x49b40821 | 32. | 0x8d2a4c8a | 48. | 0xc4ac5665 | 64. | 0xeb86d391 |

***Крок 5: вихід***

Після обробки всіх *L* 512-бітових блоків виходом *L*-ої стадії є 128-бітовий профіль повідомлення.

В цілому роботу алгоритму [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary) можна описати наступними формулами:

,

,

,

де *IV* – початкове значення буфера ABCD, визначене на кроці 3,

*Yq* – *q*-ий 512-бітовий блок повідомлення,

*L* – кількість блоків у повідомленні (включаючи біти доповнення і довжини),

*MD* – остаточне значення профілю повідомлення.

***Функція стиснення***[***MD5***](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary)

Розглянемо більш детально логіку кожного із чотирьох циклів обробки одного 512-бітного блоку. Кожний цикл складається з 16 раундів, що оперують із буфером ABCD. Кожний раунд можна представити у вигляді:

,

де A, B, C, D – чотири слова буфера; після виконання кожного окремого кроку відбувається [циклічний зсув](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187827&displayformat=dictionary) вліво на одне слово,

*f* – одна з елементарних функцій *fF, fG, fH, fI*,

*CLSS* – [циклічний зсув](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187827&displayformat=dictionary) вліво на *S* біт 32-бітового аргументу,

*X*[*k*] – *k*-те 32-бітове слово в *q*-му 512-бітовому блоці повідомлення, тобто *M*[*q*16+*k*],

*T*[*i*] – *i*-те 32-бітове слово в матриці *T*,

+ – додавання по модулю 232.

Структура операції раунду показана на рис. 4. Порядок, в якому використовуються чотири слова (A, B, C, D), забезпечує на рівні слів [циклічний зсув](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187827&displayformat=dictionary) вліво на одне слово на кожному раунді.

Рис. 4. Логіка виконання окремого раунду

На кожному із чотирьох циклів алгоритму використовується одна із чотирьох елементарних логічних функцій. Кожна елементарна функція одержує на вхід три 32-бітних слова, а на виході створює одне 32-бітне слово. Кожна функція виконує ряд побітових логічних операцій, тобто *n*-ий біт виходу є функцією від *n*-их бітів трьох вхідних значень. Елементарні функції мають наступний вигляд:

,

,

Тут бітові логічні операції AND, OR, NOT, [XOR](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187752&displayformat=dictionary) представлені символами  відповідно.

На рис. 4 показано алгоритм виконання окремого раунду. Обробляється масив 32-бітових слів *X*[0…15], що містить значення поточного 512-бітного вхідного блоку. У рамках раунду кожне з 16 слів *X*[0…15] застосовується рівно один раз, для одного кроку, а порядок, у якому ці слова використаються, міняється від циклу до циклу. В першому циклі слова фігурують в їх вихідному порядку. Для циклів з 2-го по 4-ий визначаються наступні перестановки[[1]](http://vns.lp.edu.ua/mod/page/view.php?id=189884" \l "_ftn1" \o "):

,

,

.

Кожен з 64 елементів *T* довжиною в 32-бітове слово використовується тільки один раз, для одного раунду одного циклу. Зверніть увагу й на те, що в результаті виконання одного раунду оновлюється тільки 4 байти буфера ABCD. Отже, протягом циклу кожен байт буфера оновлюється чотири рази і ще один раз наприкінці, коли отримується кінцеве вихідне значення блоку. Нарешті зазначимо, що в кожному циклі використовуються різні послідовності чотирьох циклічних зсувів уліво з різними значеннями зсуву (послідовності значень зсуву S для кожного циклу наведено в табл. 2). Під час виконання кожного циклу використовується послідовність з чотирьох значень зсувів *Sij*, яка повторюється чотири рази, фактично при обробці однакового слова буфера ABCD в одному циклі використовується однакове значення зсуву. Метою усіх цих маніпуляцій є забезпечення практичної неможливості відшукати колізії (пари 512-бітових блоків, що породжують однаковий вихід).

Таблиця 2. Значення циклічного зсуву S для кожного циклу

|  |  |  |  |
| --- | --- | --- | --- |
| **1-й цикл** | **2-й цикл** | **3-й цикл** | **4-й цикл** |
| S11=7 | S21=5 | S31=4 | S41=6 |
| S12=12 | S22=9 | S32=11 | S42=10 |
| S13=17 | S23=14 | S33=16 | S43=15 |
| S14=22 | S24=20 | S34=23 | S44=21 |

**Завдання до виконання роботи**

Створити програмну реалізацію алгоритму хешування [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary). [Тестування](http://vns.lp.edu.ua/mod/quiz/view.php?id=194686) створеної програми провести з використання наступних тестових значень хешу (згідно RFC 1321):

H() = D41D8CD98F00B204E9800998ECF8427E

H(a) = 0CC175B9C0F1B6A831C399E269772661

H(abc) = 900150983CD24FB0D6963F7D28E17F72

H(message digest) = F96B697D7CB7938D525A2F31AAF161D0

H(abcdefghijklmnopqrstuvwxyz) = C3FCD3D76192E4007DFB496CCA67E13B

H(ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789) = D174AB98D277D9F5A5611C2C9F419D9F

H(12345678901234567890123456789012345678901234567890123456789012345678901234567890) = 57EDF4A22BE3C955AC49DA2E2107B67A

Програмна реалізація повинна виводити значення хешу як для рядка, заданого в полі вводу, так і для файлу. Результат роботи програми повинен відображатись на екрані з можливістю наступного запису в файл. Крім того програма повинна мати можливість перевірити [цілісність](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187826&displayformat=dictionary) будь-якого файлу за наявним файлом з [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary) хешем, записаним у шістнадцятковому форматі. У звіті навести протокол [тестування](http://vns.lp.edu.ua/mod/quiz/view.php?id=194686) і роботи програми та зробити висновки.

**Контрольні запитання**

1.     Що таке хеш функція?

2.     Області використання хеш функцій.

3.     Назвіть вимоги до криптографічних хеш функцій.

4.     Що таке [слабка опірність колізіям](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187818&displayformat=dictionary)?

5.     Основні області використання алгоритму [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary).

6.     Яка довжина хеш-значення алгоритму [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary)?

7.     Яка довжина блоку, що обробляється алгоритмом [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary)?