УДК 681.322

С. М. Білан, І. М. Шварц

Підвищення криптостійкості та швидкодії алгоритму Blowfish у каналах передачі даних

Розглянуто можливість використання алгоритму Blowfish з метою передачі інформації каналами зв'язку як у межах приватних корпоративних мереж, так і у глобальній мережі. Запропоновано методику вдосконалення даного алгоритму та його реалізацію у вигляді програми, яка є Mapi-клієнтом поштового серверу Exchange.

Ключові слова: *захист інформації, криптографічний алгоритм, поштовий сервер, шифрування.*

Актуальність проблеми захисту інформації пов'язана зі зростанням можливостей обчислювальної техніки. Розвиток засобів, методів і форм автоматизації процесів обробки інформації та масове застосування персональних комп'ютерів роблять інформацію дуже вразливою.

Державна інформація представляє великий інтерес для кримінальних елементів. Сьогодні у керівництва більшості державних організацій немає сумнівів щодо необхідності серйозно піклуватися про інформаційну безпеку (у збереженні державних таємниць, забезпеченні безпеки електронних документів). Застосування сучасних інформаційних технологій у державних системах розширює можливості для різних зловживань, пов'язаних із використанням обчислювальної техніки (так званих комп'ютерних злочинів).

Щорічні втрати від злочинів у цій сфері складають у світі, за різними оцінками, від 170 млн. до 10 млрд. дол. За деякими даними, в промислово розвинених країнах середній збиток від одного комп'ютерного злочину (а значну частку таких злочинів складають зловживання в фінансовій сфері) близький до 450 тис. дол., а щорічні сумарні втрати в США і Західній Європі досягають 100 млрд. і 35 млрд. дол. відповідно [1]. В останні десятиріччя зберігалася стійка тенденція до зростання збитків, пов'язаних з комп'ютерною злочинністю і в Україні.

Для протидії комп'ютерним злочинам або зменшення збитку від них необхідно грамотно вибирати заходи і засоби забезпечення захисту інформації від навмисного руйнування, крадіжки і несанкціонованого доступу. Тому у нашому випадку будемо використовувати блочний шифр із секретним ключем, а саме криптографічний алгоритм Blowfish. Він оптимізований для тих задач, у яких немає час-

тої зміни ключів, таких, як поштовий клієнт, а також є мережею Фейстела (Feistel), у якої кількість ітерацій становить 16. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину у межах 448 біт [2]. Перед початком будь-якого шифрування виконується складна фаза, і алгоритм складається з двох частин: розширення ключа та шифрування даних. Розширення ключа перетворює ключ довжиною щонайменше 448 біт у кілька масивів підключів загальною довжиною 4168 байт [2].

В основі алгоритму покладено мережу Фейстела (Feistel) з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, та підстановки, що залежить від ключа та даних. Операціями є XOR та складання 32-бітних слів. Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних. Саме шифрування відбувається достатньо швидко [3].

У даній статті ставиться задача підвищення надійності та швидкості при передачі інформації комп'ютерними мережами з використанням модифікованого криптографічного алгоритму шифрування Blowfish та програмна реалізація цього алгоритму у вигляді Марі-клієнта поштового сервера.

Алгоритм шифрування

Данні шифруються на 32-бітових мікропроцесорах, і після ініціалізації у пам'яті розміщуються два масиви, якими шифруються дані.

Алгоритм використовує тільки прості операції: додавання, XOR та вибірку з таблиці по 32-бітному операнду. Аналіз його схеми не складний, що зменшує кількість помилок при реалізації алгоритму.

Довжина ключа є змінною та може досягати 448 біт.

Ключ для шифрування створюється за допомогою двох частин: перша — пароль-масив із символів; друга — час, який переводиться в комп'ютерний формат. У результаті отримуємо два 32-бітних числа, що беруть участь в ініціалізації ключа (рис. 1).

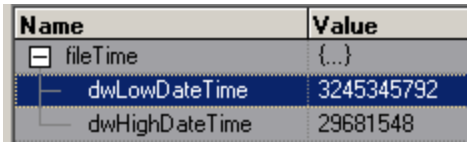

Рис. 1. Два 32-бітних числа

Алгоритм оптимізований для тих задач, в яких немає частої зміни ключів, таких, як поштовий клієнт. Данні кодуються по 32 біта, і тому у 32-розрядній системі алгоритм працює дуже швидко, набагато швидше за DES. Він не придатний для використання у задачах з частою зміною ключів, наприклад, при комутації пакетів, чи для використання у якості єдино спрямованої хеш-функції [4].

Ключ є 64-бітним блочним шифром з ключем змінної довжини. Алгоритм складається з двох частин: розгортання ключа та шифрування даних. Розгортання ключа перетворює ключ довжиною до 448 біт у кілька масивів підключів, загальним обсягом 4168 байт .

Шифрування даних базується на простій функції, що виконується послідовно 16 раз. Кожен етап складається з залежної від ключа перестановки та залежної від ключа та даних підстановки. Використовуються тільки додавання та XOR 32-

бітових слів. Єдиними додатковими операціями на кожному етапі є чотири вилучення даних з індексованого масиву.

Створення ключа

1. *P*-масив ініціалізується фіксованою строкою (рис. 2).

2. Чотири *S*-блоки ініціалізуються фіксованою строкою.

3. Виконується операція XOR P_1 з непарними 32 бітами ключа, XOR P_2 з непарними другими 32 бітами ключа та старшою 32-бітною частиною часу, і так далі для всіх бітів ключа (до P_{18}).

4. Операція 2 виконується циклічно, доки для усього *P*-масиву не буде виконана операція XOR з бітами ключа.

5. Використовуючи підключі, отримані на етапах 1, 2 і 3 алгоритмом шифрування, шифрується значення з часу. P_1 і P_2 замінюються цими результатами, а результат шифрування часу шифрується за допомогою алгоритму та змінених підключів.

6. Значення P_3 і P_4 замінюються значеннями, обчисленими у п. 4.

c->P	0x0012e860
[0]	4165234673
[1]	3761818135
[2]	56576929
[3]	2972412835
[4]	1532165121
[5]	2009599660
[6]	2052070073
[7]	3759878850
[8]	3643748150
[9]	2855859498
[10]	211628703
[11]	229215014
[12]	2975195587
[13]	1222433681
[14]	3647362257
[15]	3888116358
[16]	1378648790
[17]	3763196346

Рис. 2. Подання *P*-масиву

7. Під час процесу усі елементи *P*-масиву і потім поступово усі чотири *S*-блоки замінюються виходом алгоритму, що постійно змінюється.

Підключі зберігаються для кодування чи декодування даних.

Шифрування (дешифрування) даних

Дані повинні бути кратні 4 байтам, тому, якщо ця умова не виконується, додаємо додаткові нульові біти до розміру, який кратний чотирьом. Наприклад, *P*-масив дорівнює 12 байт, або три 32-бітних числа (рис. 3).

У функцію шифрування передаються два 32-бітних числа, обираються спочатку перше і останнє, поступово направляючись до середини:

1,2,3,4,…20,21,22…38,39,40,41.

pBuf,12	0x00e9aa70 ""
[0]	0 ''
[1]	0 ''
[2]	0 ''
[3]	0 ''
[4]	6 'Ъ'
[5]	207 'П'
[6]	240 'р'
[7]	232 'и'
[8]	226 'в'
[9]	179 'і'
[10]	242 'т'
[11]	0 ''

Рис. 3. Вхідна інформація, або три 32-бітних числа

Якщо залишається одне число, то воно кодується разом з наступним елементом масиву після нього. Так, у наведеному прикладі, 21 буде кодуватися з 41. При обраній операції ці числа будуть кодуватися першими. Це дає можливість кодувати як парну кількість подвійних слів, так і непарну.

data,3	0x00e9ade8
[0]	1427316382
[1]	3983571191
[2]	404923365

Рис. 4. Вихідна інформація або три 32-бітних числа

Процес кодування двох 32-бітних чисел X_L, і X_R складається з наступних етапів (рис. 4).

1. Послідуючі операції виконуються у циклі 16 раз.

1.1. $x_L = x_L \oplus x_R$ — числа додаються за модулем 2.

Строка коду: «X1 = X1 ^ c–>P[*i*]».

1.2. x_R = Func(x_L) $\oplus x_R$ — число x_R додається за модулем 2 з $F(x_L)$, де $F(x_L)$ — це функція, яка виконує наступні дії.

1.2.1. Розділяємо x_L на чотири 8-бітних частини: *a*, *b*, *c*, *d*, що здійснюється зсувом цього числа праворуч на 8 біт (рис. 5).

1.2.2. Функція обчислюється як $((S_{1,a} + S_{2,b} \bmod 2^{32}) \oplus S_{3,c}) + S_{4,d} \bmod 2^{32}$.

Строка коду: «Xr = Func(c, X1) ^ Xr».

1.3. Переставляємо x_L та x_R.

Строки коду: «temp = X1; X1 = Xr; Xr = temp».

2. Переставляємо x_L та x_R: «temp = X1; X1 = Xr; Xr = temp».

3. $x_R = x_R \oplus P_{17}$ — числа додаються за модулем 2 з елементом ключа P_{17}.

Строки коду: «Xr = Xr ^ c–>P[N]».

4. $x_R = x_R \oplus P_{18}$ — число додається за модулем 2 з елементом ключа P_{18}.

Строки коду: «X1 = X1 ^ c–>P[N + 1]».

Ми отримали закодоване повідомлення, а результат в 32-бітних числах.

Name	Value
pBuf,12	0x00e9ade8 "ћЂЂUчхрнеJ"Ъзззз««««««««««««
[0]	158 'ћ'
[1]	30 'Ђ'
[2]	19 'Ђ'
[3]	85 'U'
[4]	247 'ч'
[5]	120 'х'
[6]	112 'р'
[7]	237 'н'
[8]	229 'е'
[9]	163 'J'
[10]	34 '"'
[11]	24 'Ъ'

Рис. 5. Вихідна інформація у 8-бітних значеннях

Дешифрування виконується аналогічно шифруванню, але P_1, P_2, ..., P_{18} використовуються у зворотному порядку.

Даний алгоритм був змодульований програмними засобами Mathcad (рис. 6).

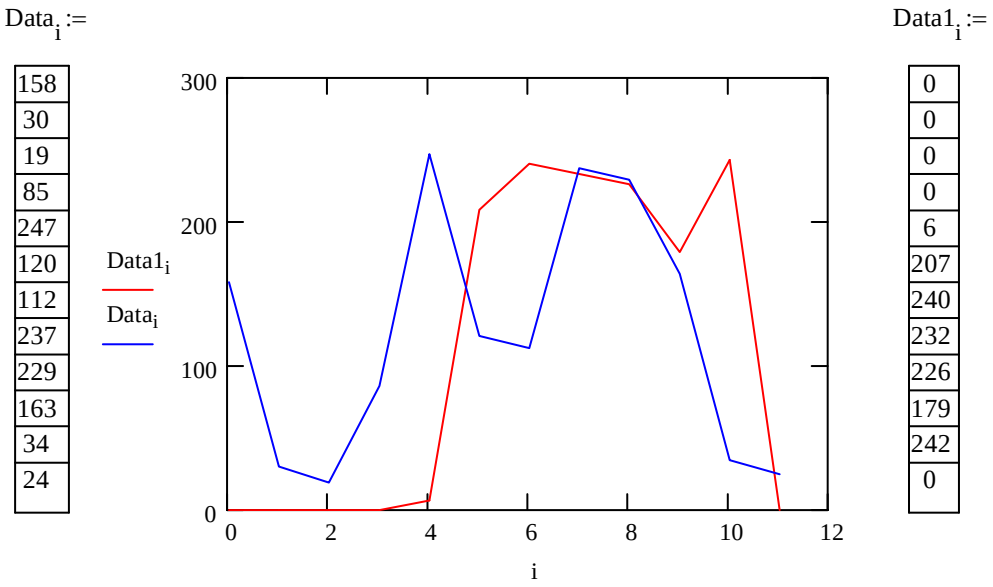

Рис. 6. Графік залежності вхідної та вихідної інформації

Як бачимо з графіка, зашифровані дані не мають лінійних властивостей. Це вказує на те, що в даному криптографічному методі не має чітко виражених логічних дій, що в свою чергу, свідчить про криптографічну стійкість цього методу.

Для слабких ключів, які генерують погані *S*-блоки, додавання двох додаткових 32-бітних ключів (з часу) підвищує захищеність ключа, що підвищує якість шифрування. При невідомих *S*-блоках ми можемо виявити використання слабкого ключа, але не можемо визначити сам ключ (*S*-блоки та *P*-масив). Тому цей метод ефективний тільки проти варіантів зі зменшеною кількістю етапів та абсолютно недієвий проти 16-етапного.

Короткий опис програми

Підготовка до пересилання даних на поштовий сервер складається з таких етапів:

1) створюється ключ кодування: перший ключ — пароль доступу до пошти, другий — час створення повідомлення;

2) дані, що містяться у повідомленні, шифруються нашим алгоритмом;

3) вкладені дані завантажуються у пам’ять і сереалізуються в архів, у результаті якого отримуємо байтовий масив;

4) байтовий масив шифрується за допомогою алгоритму кодування як 32-бітний (якщо кількість байт у масиві не кратна 4 байтам, то масив розширюється до потрібного розміру);

5) усе повідомлення перетворюється у 7-бітний код за допомогою алгоритму base 64 та відправляється [5];

6) при отриманні перевіряється, кому відправлене кожне повідомлення, воно розшифровується у зворотній послідовності;

7) усі дані у скриньках користувача зберігаються у зашифрованому вигляді.

Висновки

У даній статті був проаналізований криптографічний метод Blowfish, запропоновано спосіб його вдосконалення, що підвищує загальну криптостійкість системи. Цей метод був впроваджений у поштовій програмі, яка є клієнтом поштового серверу Microsoft Exchange. Розроблена програма дає можливість не тільки обмінюватися секретними даними електронною поштою, використовуючи принципово новий метод їх шифрування, але й введення власних користувачів, як з одного боку мережі, так і з іншого, тобто дає можливість умовного поділу однієї поштової скриньки та використання її багатьма користувачами з власними паролями доступу. Основними перевагами запропонованого алгоритму є поліпшена крипостійкість за рахунок створення більш надійного ключа, а також підвищення швидкодії за рахунок шифрування по два 32-бітних числа.

1. Организация и современные методы защиты информации. — М.: Концерн «Банковский Деловой центр», 1998. — 465 с.

2. *Брюс Шнайер.* Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С. 2-е издание. — М.: Дело, 2003. — 524 с.

3. *Smith J.L.* The Design of Lucifer. A Cryptographic Device for Data Communication, RC 3326, White Plains: IBM Research.

4. *Горбенко И.Д., Долгов В.И., Рублинецкий В.И.* Мифы и реальность // Безопасность информации. — 1996. — № 2. — С. 17–25.

5. *Biham E. and Shamir A.* Differential Cryptanalysis of the Data Encryption Standard. — Springer-Verlag, 1993.

Надійшла до редакції 18.01.2005