УДК 681.3.06

АНАЛІЗ БЕЗПЕКИ MAC АЛГОРИТМІВ СТАНДАРТУ ISO/IEC 9797-2

Геннадій Халімов, Олександр Потій, Олексій Дунь, Євген Котух
Харківський національний університет радіоелектроніки

Анотація: ***Розглянути MAC-алгоритми, що визначені у стандарті ISO/IEC 9797-2, проведено аналіз їх безпечності до основних атак.***
Summary: ***MAC-algorithms which are certain in the standard ISO/IEC 9797-2 are considered, the analysis from safety to the basic attacks is lead.***
Ключові слова: ***MAC алгоритм, геш-функція.***

Вступ

Відповідно до Плану стандартизації та програми гармонізації міжнародних стандартів в галузі криптографічного захисту інформації в Україні здійснюються роботи щодо вивчення та аналізу міжнародних стандартів. Важливою задачею є формування рекомендацій щодо можливості та особливостей застосування міжнародних стандартів в сфері КЗІ в Україні. Тому актуальними є дослідження криптографічних примітивів та механізмів, що стандартизуються в ISO. У цій роботі викладаються результати аналізу механізмів формування кодів автентифікації повідомлень (MAC), що стандартизуються у міжнародному стандарті ISO/IEC 9797. Три механізми, що визначені у стандарті ISO/IEC 9797-2 [1], базуються на спеціалізованих геш-функціях RIPEMD-160 , RIPEMD-128 та SHA-1, які описано в ISO/IEC 10118-3 [2]. Перший механізм, який описано в цьому стандарті, відомий як MDx-MAC (алгоритм 1). MDx-MAC викликає повну геш-функцію один раз, проте вносить деякі незначні зміни до циклової функції шляхом додавання ключа до адитивних констант циклової функції. Другий механізм, який описано в цьому стандарті, відомий як HMAC (алгоритм 2). HMAC викликає повну геш-функцію два рази. Третій механізм, який описано в цьому стандарті є модифікацією MDx-MAC (алгоритм 3), що бере як вхідні дані лише короткі рядки (щонайбільше 256 бітів). Він забезпечує більшу продуктивність для додатків, що оперують лише з короткими рядками.

Вимоги стандарту ISO/IEC 9797-2 визначають, що користувачі мають обрати:

- MAC алгоритм з тих, що описані у стандарті;
- спеціалізовану геш-функцію з тих, що описані у ISO/IEC 10118-3;
- довжину m MAC (у бітах).

Довжина m MAC для алгоритмів 1 та 2 має бути додатним цілим, меншим або рівним довжині геш-коду L_H. Довжина m MAC для алгоритму 3, має бути додатним цілим, меншим або рівним половині довжині геш-коду, тобто $m \geq L_H / 2$. Довжина (у бітах) рядка даних D має бути щонайбільше $2^{64} - 1$ для алгоритмів 1 та 2, та щонайбільше 256 для алгоритму 3.

Зроблений вибір впливає на рівень безпечності алгоритму.

Для обчислення та перевіряння MAC має використовуватися один ключ. Якщо рядок даних також зашифровується, ключ обчислення MAC повинен відрізнятися від ключа зашифровування.

Завданням статті є аналіз загальних положень стандарту ISO/IEC 9797-2, опис алгоритмів формування MAC та аналіз їх безпечності.

I Алгоритм MDx-MAC

Алгоритм 1 при обчисленні значення MAC однократно застосовує геш-функцію. Геш-функція обирається зі спеціалізованих геш-функцій RIPEMD-160 , RIPEMD-128, SHA-1 з ISO/IEC 10118-3:1998. Розмір ключа K (у бітах) має бути щонайбільше 128 бітів. Алгоритм MDx-MAC, відповідно до застосовуваної геш-функції, має визначення RIPEMD-160 -MAC, RIPEMD-128-MAC, або SHA-1-MAC.

Алгоритм 1 містить п’ять кроків: розгортання ключа, модифікації констант та вектору ініціалізації IV, операції гешування, результуючого перетворення та усікання. Структурна схема алгоритму подана на рис. 1.

Розгортання ключа здійснюється в разі, якщо K коротше ніж 128 бітів. Для формування 128 бітового ключа K' необхідно взяти 128 лівих бітів рядка із декілька разів повтореного первинного ключа (якщо

довжина K (у бітах) дорівнює 128, то $K' := K$):

$$K' := 128 \sim (K \| K \| \ldots \| K).$$

Далі обчислюються підключі K_0, K_1 та K_2:

$$K_0 := \bar{h}(K' \| U_0 \| K')$$
$$K_1 := 128 \sim \bar{h}(K' \| U_1 \| K')$$
$$K_2 := 128 \sim \bar{h}(K' \| U_2 \| K'),$$

де U_0, U_1 та U_2 – 768 бітові константи, які визначені у стандарті, $\bar{h}$ – означає спрощену геш-функцію h, без заповнення та додавання довжини.

Вироблений ключ K_1 розбивається на чотири 32 бітних слова $K_1 = K_1[0] \| K_1[1] \| K_1[2] \| K_1[3]$.

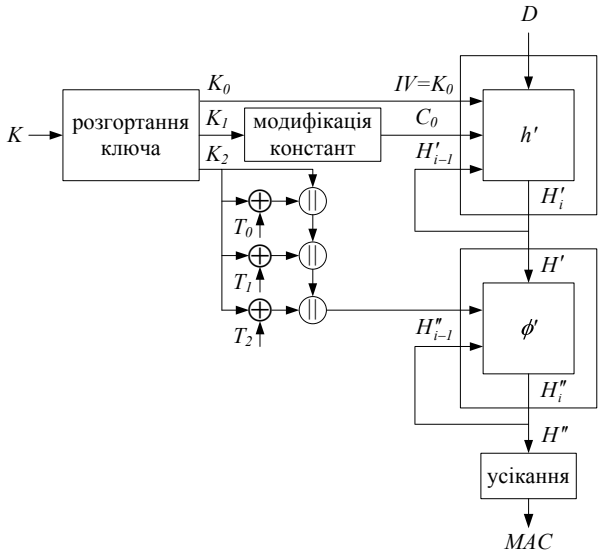

Рисунок 1 – Схема алгоритму 1

Модифікація констант, геш-функція IV ***та операція гешування.*** Адитивні константи, що використовуються в цикловій функції, модифікуються додавання за модулем 2^{32} одного з чотирьох слів K_1, наприклад $C_0 := C_0 \boxplus K_1[0]$.

Початкове значення геш-функції IV замінюється в такий спосіб $IV := K_0$. Результуюча геш-функція позначається як h', а її циклічна функція позначається як ϕ'.

Рядок, що слугує вхідними даними для модифікованої геш-функції h' є еквівалентним з рядком даних D, тобто $H' = h'(D)$.

Результуюче перетворення та усікання. Результуюче перетворення застосовує модифіковану циклічну функцію φ', першими вхідними даними є рядок $K_2 \| (K_2 \oplus T_0) \| (K_2 \oplus T_1) \| (K_2 \oplus T_2)$, а другими вхідними даними – рядок H', тобто:

$$H'' := \phi'(K_2 \| (K_2 \oplus T_0) \| (K_2 \oplus T_1) \| (K_2 \oplus T_2), H').$$

Значення T_0, T_1 та T_2 – це 128 бітові рядки, визначені у стандарті для кожної спеціалізованої геш-функції.

MAC, що складається з m бітів виробляється шляхом взяття m лівих бітів рядка H'', тобто:

$$MAC := m \sim H''.$$

Розглянемо ефективність MDx-MAC алгоритму. Алгоритм 1 потребує $q+7$ застосувань циклічної функції, якщо заповнений рядок даних складається з q блоків. Кількість застосувань може бути знижено до $q+1$ шляхом попереднього обчислення значень K_0, K_1, K_2 та заміною ініціюючих значень IV та IV' в застосуванні геш-функції. Для довгих вхідних рядків MAC алгоритм 1 має ефективність, порівнянну з тією, що має використовувана геш-функція.

Аналіз безпечності алгоритму 1 (MDx-MAC) виконано в [3]. Розглянемо результати аналізу.

Атака вгадування MAC для алгоритму 1 є загальною та має імовірність успіху $\max\left(1/2^m, 1/2^K\right)$. Протистояти цій атаці можна лише шляхом обґрунтованого виробу параметрів m та K.

Атака відновлення ключа типу груба сила для MAC алгоритму 1 також є загальною і в середньому потребує виконання 2^{K-1} операцій; перевірка такої атаки потребує приблизно K/m пар «рядок даних – MAC». Протистояти цій атаці можливо шляхом обґрунтованого виробу ключа K. Альтернативним шляхом протистояння є запобігання можливості будь-кому отримати необхідні для ідентифікації ключа K/m пар «рядок даних – MAC». Наприклад, якщо $k^* = 64$ та $m = 32$, то приблизно 2^{32} ключів відповідає заданій парі «рядок даних – MAC»; якщо ключі змінюють після кожного рядка даних, атака відновлення ключа типу груба сила є не більш ефективною за вгадування значення MAC.

Атака, що заснована на «парадоксі дня народження» визначається у [4] і потребує одного обраного рядка даних та приблизно $2^{n/2}$ відомих рядків даних та 2^{n-m} обраних рядків даних. Для SHA-MAC при $n = 2m = 160$ біт атака підробки потребує приблизно 2^{80} обраних текстів і 2^{80} відомих текстів. Відповідне число відомих та обраних текстів для MD5-MAC та RIPEMD-MAC становить приблизно 2^{64}.

Найкращим атаками є атаки, визначені у пропозиції 1 та 2.

Пропозиція 1 [3]. Якщо MDx є секретною геш-функцією MD подібної родини, то найкращою атакою на MDx-MAC є атака вичерпного пошуку (відносно ключового відновлення), і атака за пропозицією 2 (відносно підробки MAC коду).

Пропозиція 2 [3]. Нехай h є ітераційним MAC із n бітовою модифікуємою змінною, m бітовим результатом, функцією стискання f, яка є випадковою функцією (для фіксованого значення блоку даних x_i) і результуючим перетворенням g. Внутрішню колізію для h може бути створено на основі u відомих пар «текст-MAC», де кожен текст є деяким підрядком з $s \geq 0$ пов'язаних блоків та v обраних текстів. Значення для u та v визначаються у такий спосіб: $u = \sqrt{2/(s+1)} \cdot 2^{n/2}$; $v = 0$, якщо перетворення g є перестановкою та $s+1 \geq 2^{n-m+6}$ (можливе число зовнішніх колізій є малим). Якщо перетворення g є випадковою функцією, то v приблизно дорівнює

$$2\left(\frac{2^{n-m}}{s+1}\left(1-\frac{1}{e}\right)+\left\lfloor\frac{n-1-\log_2(s+1)}{m-1}\right\rfloor\right).$$

Якщо бітовий розмір внутрішньої змінної алгоритму, що модифікується, дорівнює подвоєному значенню MAC результату, що рекомендується, (тобто $n = 2m$), то атака підробки вимагає $O\left(2^m/(s+1)\right)$ відомих пар «текст–MAC» та $O\left(2^m/\sqrt{(s+1)}\right)$ обраних текстів. Для випадку $m = n$ MDx-MAC алгоритм є кращим, ніж алгоритм 2 (HMAC).

Для MDx-MAC не знайдено атаки ключового відновлення на основі методу декомпозиції. Роль змінних K_0 та K_1 у алгоритмі 1 є подібною до ролі секретних ключів K_0 та K_1 у алгоритмі 2 (HMAC), але з тією відмінністю, що атака методом декомпозиції, як доведено в [3], не має успіху. Використання K_1 в MDx-MAC порівняно з HMAC забезпечує додатковий захист у випадку слабості сучасних геш-функцій. Вичерпний пошук для K_i так само важкий, як вичерпний пошук для K.

Передбачається, що MDx-MAC є також стійким проти атак, заснованих на слабостях внутрішньої структури MDx. Перевага алгоритму 1 полягає в мінімізації відмінностей між MDx та MDx-MAC, що знижує ймовірність впливу нових слабкостей геш-функцій, які можуть бути виявлені пізніше. Наприклад, як було показано в [5], функція стискання MD5 не є колізійно стійкою, проте цей результат не говорить про слабкість MD5-MAC.

II Алгоритм HMAC

Алгоритм 2 (HMAC) для обчислення значення MAC потребує однократного застосування геш-функції. Геш-функція обирається з ISO/IEC 10118-3 таким чином, щоб L_1 було додатним цілим кратним 8 та $L_2 \le L_1$, де L_2 – це розмір геш-коду (у бітах), L_1 – розмір (у бітах) вхідного рядка даних до циклічної функції. Розмір ключа K (у бітах) має бути щонайменше L_2, тобто $L_2 \le K \le L_1$.

HMAC алгоритм потребує п’ять кроків: розгортання ключа, операції гешування, результуючого перетворення та усікання. Структурна схема алгоритму подана на рис.2.

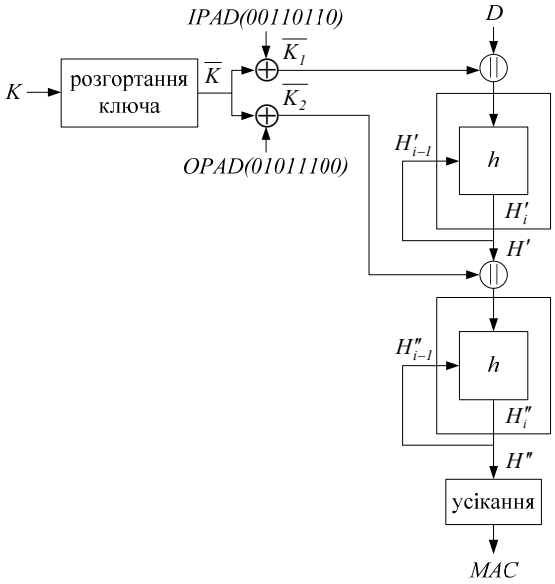

Рисунок 2 – Схема MAC алгоритму 2

Розгортання ключа та хешування. Розгортання ключа включає додавання ($L_1 - K$) нульових бітів праворуч до ключа K. Результуючий рядок дожини L_1 позначається як $\overline{K}$.

Ключ $\overline{K}$ розгортається у два підключа $\overline{K}_1$ та $\overline{K}_2$. Перший підключ $\overline{K}_1$ формується за виразом $\overline{K}_1 = \overline{K} \oplus IPAD$, де рядок $IPAD$ є поєднання $L_1/8$ разів значення ‘36’ в шістнадцятковому поданні (бінарне значення ‘00110110’).

Другий підключ $\overline{K}_2$ визначається виразом $\overline{K}_2 = \overline{K} \oplus OPAD$, де рядок $OPAD$ є поєднання $L_1/8$ разів значення ‘5C’ в шістнадцятковому поданні (бінарне значення ‘01011100’).

Гешування виконується над рядком, що є поєднанням $\overline{K}_1$ та D

$$H^{'} = h\left(\overline{K_1} \,||\, D\right).$$

Результуюче перетворення та усікання. Результуюче перетворення виконується над рядком, що є поєднанням $\overline{K}_2$ та $H^{'}$

$$H^{''} = h\left(\overline{K_2} \,||\, H^{'}\right).$$

MAC, що складається з m бітів, виробляється шляхом взяття m лівих бітів рядка $H^{"}$, тобто:

$$MAC := m \sim H^{"}.$$

Розглянемо ефективність алгоритму. HMAC алгоритм потребує $q+3$ застосувань циклічної функції, якщо заповнений рядок даних складається з q блоків. Кількість застосувань може бути знижено до $q+1$ шляхом внесення змін до коду геш-функції. Можна заздалегідь обчислити значення $IV_1 := \phi\left(\overline{K_1}, IV\right)$ та

$IV_2 := \phi\left(\overline{K_2}, IV\right)$, замінити ініціююче значення IV на IV_1 при першому застосуванні геш-функції та на IV_2 у результуючому перетворенні (друге застосування геш-функції). Це може вимагати внесення модифікації до методу заповнення: дійсно, тепер вхідний параметр геш-функції на L_1 бітів коротше; це означає, що значення L_1 тепер має додаватися до значення L_D.

Для довгих вхідних рядків алгоритм 2 має ефективність, порівняну з тією, що має використовувана геш-функція.

Аналіз безпечності алгоритму 2 (HMAC), ще відомий як метод обгортки (envelope method), виконано в [3]. Метод обгортки є комбінацією префікс і суфікс методів формування MAC.

Атака вгадування MAC для алгоритму 2 та **атака відновлення ключа типу груба сила** є загальними та мають, відповідно, імовірність успіху $max\left(1/2^m, 1/2^K\right)$, та 2^{K-1} операцій перевірянь, приблизно K/m пар «рядок даних–MAC». **Атака, що заснована на «парадоксі дня народження»** потребує одного обраного рядка даних, приблизно $2^{n/2}$ відомих рядків даних та 2^{n-m} обраних рядків даних.

В [3] показано, що існує атака ключового відновлення методом декомпозиції – на відміну від результатів, наданих в [6], де стверджується, що атака методом декомпозиції вимагає вичерпного пошуку для ключа $K_1 + K_2$ біт. Схема нової атаки полягає у такому. Зловмисник може утворити внутрішню колізію для ланцюгової змінної алгоритму 2. Можна виконати вичерпний пошук ключа K_1, крім всіх випробуваних значень ключів, які не дають внутрішню колізію, що потребує 2^{K_1} операцій поза лінією. Для знаходження K_1 потрібно трохи більше ніж K_1/n перевірок. Після цього метод обгортки переводиться в суфікс метод із секретним ключем K_2. Оскільки K_1 визначено, то для визначення K_2 може бути використано вичерпний пошук. Зауважимо, що вибір $K_1 \neq K_2$ не додає істотного підвищення безпечності MAC алгоритму порівняно з рівністю $K_1 = K_2$. Проте атака на алгоритм все ж таки вимагає великої кількості відомих пар «текст-MAC».

Підробка MAC, як наслідок пропозиції 2, за умови $m = n = 128$ та $s \geq 1$, припускаючи, що останній блок містить тільки значення ключа K_2, вимагатиме $2^{56,5}$ відомих пар «текст-MAC» та один обраний текст за умовою, що $s = 2^{16}$. Цей результат показує, що алгоритм 2 значно слабший, ніж ідеальний MAC за припущення ключового розміру $K_1 + K_2$.

Значення таємності MAC алгоритмів для випадку $n = m$ надані в табл. 1 [3]. Аналіз табл. 1 показує, що при застосуванні методу обгортки довжина K_1 не має бути занадто малою.

Таблиця 1 – Безпечність MAC алгоритмів

	Ідеальний MAC		Префікс метод		Суфікс метод		Метод обгортки (HMAC)	
	#MAC	*#op*	*#MAC*	*#op*	*#MAC*	*#op*	*#MAC*	*#op*
Ключове відновлення	$\left\lceil \frac{k}{n} \right\rceil$	2^k	1	0	$\left\lceil \frac{k_2}{n} \right\rceil$	2^{k_2}	$\left\lceil \frac{k_1 + k_2}{n} \right\rceil$ $2^{n/2}$	$2^{k_1+k_2}$ $2^{k_1} + 2^{k_2}$
MAC підробка	$\left\lceil \frac{k}{n} \right\rceil$	2^k	1	1	1	$2^{n/2}$	$5C + 2^{n/2}$ $2^{n/2}$	0 2^{k_1}
Атака другого прообразу	2^n	0	t	$2^n/t$	t	$2^n/t$	2^n	0

В табл. 1 прийнято такі позначення: *#MAC* – число відомих пар «текст-MAC», *C* – число обраних текстів, *#op* – число MAC обчислень поза лінією, число повідомлень або блоків k_1, k_2, k_3, наданих криптоаналітику.

III Алгоритм 3

Алгоритм 3 оптимізовано для коротких вхідних даних (щонайбільше 256 бітів). Геш-функція має обиратися з спеціалізованих геш-функцій з ISO/IEC 10118-3:1998.

Розмір ключа K (у бітах) має бути щонайбільше 128 бітів, а довжина MAC m (у бітах) щонайбільше $L_H / 2$.

Алгоритм 3 потребує п’ять кроків: розгортання ключа, модифікації констант циклової функції, заповнення, застосування циклової функції та усікання. Структурна схема алгоритму подана на рис. 3.

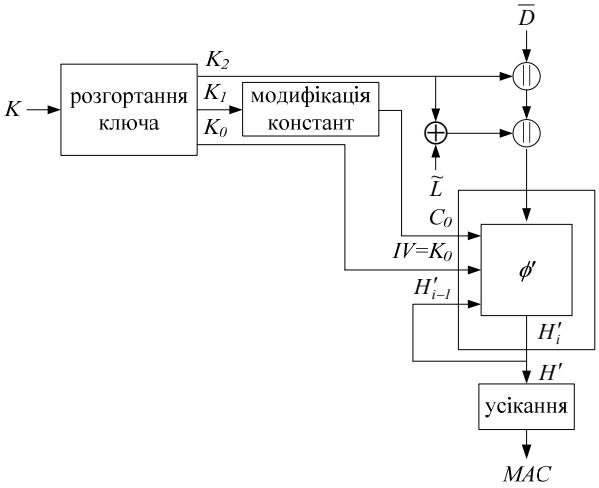

Рисунок 3 – Схема алгоритму 3

Розгортання ключа виконується, якщо K коротше ніж 128 бітів, шляхом поєднання його з самим собою необхідною кількістю разів. Для формування 128 бітового ключа $K^{'}$ необхідно взяти 128 лівих бітів (у випадку якщо довжина K (у бітах) дорівнює 128, то $K^{'} := K$):

$$K^{'} := 128 \sim \left(K \| K \| \ldots \| K\right).$$

Обчислюються підключі K_0, K_1 та K_2:

$$K_0 := \bar{h}\left(K^{'} \| U_0 \| K^{'}\right),$$
$$K_1 := 128 \sim \bar{h}\left(K^{'} \| U_1 \| K^{'}\right),$$
$$K_2 := 128 \sim \bar{h}\left(K^{'} \| U_2 \| K^{'}\right).$$

Значення U_0, U_1 та U_2 – це 768 бітові константи, які визначені в стандарті, $\bar{h}$ означає спрощену геш-функцію h, тобто без операцій заповнення та додавання довжини.

Вироблений ключ K_1 розбивається на чотири слова, що позначаються як $K_1[i]$ ($0 \le i \le 3$), тобто:

$$K_1 = K_1[0] \| K_1[1] \| K_1[2] \| K_1[3].$$

Для перетворення рядка в слова, необхідне погодження з нумерацією байтів. Для цього перетворення приймається погодження з нумерацією байтів, яку визначено для кожної спеціалізованої геш-функції в ISO/IEC 10118-3.

Модифікація констант, геш-функція IV ***та заповнення рядку даних.*** Адитивні константи, що використовуються в цикловій функції, модифікуються додаванням за модулем 2^{32} одного з чотирьох слів K_1, наприклад:

$$C_0 := C_0 \boxplus K_1[0].$$

Початкове значення геш-функції IV замінюється в такий спосіб $IV := K_0$. Результуюча геш-функція

позначається як h', а її циклова функція позначається як ϕ'.

Біти, що доповнюються до справжнього рядка даних, використовуються лише для обчислення MAC. В подальшому немає потреби у збереженні або передаванні разом із даними цих бітів (якщо вони взагалі є). Перевіряючий має знати, чи зберігаються або передаються біти, що використовувалися для заповнення.

Для того, щоб бути вхідними даними MAC алгоритму, рядок даних D має бути доповнений праворуч кількома (можливо жодним) нульовими бітами, оскільки необхідно отримати рядок даних $\overline{D}$, довжина якого становить 256 бітів.

Застосування циклової функції та усікання. Бітовий рядок $\tilde{L}$ обчислюється як двійкове подання довжини L_D (у бітах) рядка даних D, що доповнене зліва кількома «0», оскільки необхідно сформувати 128 бітовий рядок. Крайній правий біт бітового рядка $\tilde{L}$ відповідає найменшому біту бінарного подання L_D.

Рядок, що є вхідними даними циклової функції ϕ' (з модифікованими константами), дорівнює поєднанню K_2, D та результату операції виключного АБО від K_2 та $\tilde{L}$:

$$H' := \varphi'\left(K_2 \| \overline{D} \| \left(K_2 \oplus \tilde{L}\right), IV'\right).$$

MAC, що складається з m бітів, утворюється шляхом взяття m лівих бітів рядка H'', тобто:

$$MAC := m \sim H''.$$

Розглянемо ефективність та безпечність алгоритму 3. MAC алгоритм 3 потребує семиразового застосування циклової функції. Кількість застосувань може бути знижено до одного застосування циклової функції шляхом попереднього обчислення значень K_0, K_1 та K_2. Безпечність MAC алгоритму 3 подібна до припущень, які зроблені до циклової функції ϕ при доведені безпечності MAC алгоритмів 1 та 2.

Висновки

Алгоритм 1 є безпечним, якщо циклова функція ϕ, керована ініціюючим значенням геш-функції IV та адитивними константами, є псевдовипадковою функцією.

Алгоритм 2 є безпечним при виконанні таких припущень:

- геш-функція h є колізійно-стійкою у випадку збереження у таємності геш-функції IV;
- циклова функція ϕ, керована ініціюючим значенням геш-функції IV, є стійким MAC алгоритмом (тобто, його результуюче значення складно прогнозувати);
- ключі $\overline{K_1}$ та $\overline{K_2}$ не можуть бути розпізнані серед дійсно випадкових ключів; ця вимога є подібною до вимоги, що циклова функція ϕ, керована ініціюючим значенням геш-функції IV, має бути «слабкою» псевдовипадковою функцією («слабкість» полягає в тому, що зловмисник має лише побічний доступ до значень $\overline{K_1}$ та $\overline{K_2}$).

Безпечність алгоритму 3 подібна до припущень, які зроблені до циклової функції ϕ при доведенні безпечності алгоритмів 1 та 2.

*Литература: **1.** International Organization for Standardization, \ISO/IEC 9797-2: Information Technology - Security Techniques - Message Authentication Codes (MACs) – Part 2: "Mechanisms using a dedicated hash-function." 2002. **2.** International Organization for Standardization, \ISO/IEC 10118-3: Information Technology - Security Techniques - Hash-functions - Part 3: Dedicated hash-functions." 1998. **3.** B. Preneel, P. C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," Advances in Cryptology. Proceedings Crypto'95, LNCS 963, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 1 – 14. **4.** B. Preneel, P. C. van Oorschot, "On the security of iterated Message Authentication Codes." IEEE Transactions on Information Theory, Vol. 45, No. 1, January 1999, pp. 188-199. **5.** B. den Boer, A. Bosselaers, "Collisions for the compression function of MD5", Proc.Eurocrypt'93,*

LNCS 765, Springer-Verlang, 1994, pp. 293 – 304. ***6.*** *B. Preneel, Cryptographic Hash Function, Kluwer Academic Publishing, 1995(to appear).*

УДК 681.3.06

ВИКОРИСТАННЯ НЕЙРОННОЇ МЕРЕЖІ КОХОНЕНА ДЛЯ РОЗПІЗНАВАННЯ СПАМУ

Ігор Терейковський
Державний університет інформаційно-комунікаційних технологій

Аннотация: ***Досліджена можливість використання сучасних модифікацій нейронної мережі Кохонена в системах захисту електронної пошти від спаму. Проведено адаптацію архітектури нейронної мережі типу пружинна карта для кластеризації електронних листів. Обґрунтовані принципи визначення та попередньої обробки вхідних параметрів.***
Summary: ***The opportunity of use of modern modifications of the neuron network Kohonena in systems of protection of electronic mail from spama is investigated. The adaptation of architecture of neuron network of a type a spring card for clusterisation of the electronic letters is carried out. The principles of definition and initial processing of entrance parameters are developed.***
Ключевые слова: ***Електрона пошта, спам, нейронні мережі, карта Кохонена, пружинна карта.***

Електронна пошта є одним із найбільш поширених та важливих сервісів як локальних комп'ютерних мереж, так і глобальної мережі Інтернет. Її роль пояснюється тим, що вона використовується не тільки для доставки приватних повідомлень, але й як важливий компонент системи електронного документообігу практично всіх підприємств та організацій. З цієї причини надійність та безпечність функціонування електронної пошти важлива як для приватних, так і для корпоративних користувачів. При цьому задача захисту електронної пошти від спаму на сьогодні є далекою від вирішення. Відзначимо, що в загальному випадку під терміном спам звичайно розуміють масово розповсюджувані листи, зміст яких носить рекламний або шахрайський характер. Як правило такі листи анонімні, а більшість користувачів не давали своєї згоди на отримання цієї розсилки. По даним [1, 2] спам складає близько 70 – 80 % від всього обсягу електронних листів в російськомовній зоні Інтервент. Основною причиною існування спаму є впевненість багатьох комерційних установ, що це є найбільш ефективним видом реклами товарів та послуг. Крім того, все частіше в спамі зустрічаються різноманітні обманні пропозиції. Наприклад, це може бути об'ява про зміну доменної адреси електронного магазину, в якому користувач може розрахуватись за допомогою безготівкового платежу по мережі Інтернет. Зрозуміло, що адреса є фіктивною, а користувач переказавши гроші товар не отримує. Такий вид спаму отримав назву “фішінг”.

Процес розповсюдження спаму є досить прибутковим бізнесом та поставлений на професійну основу. Розроблені та широкодоступними є спеціальні програмні засоби, що дозволяють оминаючи антиспамовий захист провести розсилку декількох мільйонів спам-листів протягом 2 – 3 годин. Наприклад, на Веб-сайтах *inattack.ru та www.izone.ru* розміщені, хоча і під іншою назвою та в дещо спрощеному варіанті, інсталяційні пакети прикладних програм для розсилки спаму. Тому масовими розсилками електронних листів все частіше починають займатись не тільки професійні спамери, але й звичайні користувачі Інтернет.

Якщо не враховувати психологічний аспект проблеми, то для кінцевого користувача основним негативним наслідком отримання спаму є збільшення обсягу нецільових листів, перегляд та класифікація яких потребує значного часу. Результати [1, 2] вказують, що не зважаючи на відповідний захист співробітники великих комерційних організацій, які відповідають за електронну пошту витрачають на обробку спаму до 30% свого робочого часу. Водночас збільшується Інтернет трафік та дисковий простір на поштових серверах та робочих станціях. Це вказує на актуальність проведення досліджень в напрямку застосування нових підходів для боротьби проти спаму.

І Недоліки загальнопоширених методів розпізнавання спаму

Сучасні методи боротьби зі спамом можливо розділити на законодавчі, організаційні та програмно-технічні. Розглянемо останні. Вони функціонують за принципом: розпізнання спаму – блокування (знищення) спаму. Відзначимо, що блокування розпізнаного спаму не викликає труднощів. При цьому задача розпізнавання спаму потребує доопрацювання [1 – 3], хоча це і суперечить рекламним заявам провідних виробників антиспамових засобів. Так в [1] зазначено, що термін блокування нового виду спаму