УДК 004

Любарський С.В., Шаціло П.В.

ОРГАНІЗАЦІЯ ЗАХИЩЕНОГО ОБМІНУ SOAP-ПОВІДОМЛЕННЯМИ МІЖ ASP.NET WEB-СЕРВІСОМ ТА WEB-ДОДАТКОМ ЗА ДОПОМОГОЮ АЛГОРИТМУ БЛОЧНОГО ШИФРУВАННЯ AES

Анотація:

У статті розглядається підхід до вирішення задачі по шифруванню відкритого трафіка між WEB-клієнтом і WEB-сервісом для мереж спеціального призначення на основі стандарту AES за алгоритмом блокового симетричного шифрування.

Аннотация:

В статье рассматривается подход применительно к решению задачи по шифрованию открытого трафика между WEB-клиентом и WEB-сервисом для сетей специального назначения на основе стандарта AES по алгоритму блочного симметричного шифрования.

Abstract:

The paper considers the approach for solving the problem by encrypting the traffic between the open-WEB-client and WEB-service for special-purpose networks based on standard AES algorithm for symmetric encryption block.

Порушення безпеки інформаційної системи органу управління спеціального призначення розглядається як інцидент безпеки інформаційно-аналітичного процесу управління об'єктами управління через втрату конфіденційності, цілісності, доступності інформаційних ресурсів. Особливо актуальним постає питання забезпечення конфіденційності інформації та цілісності даних в технологіях передачі інформації, що базуються на відкритих стандартах функціонування *Web*.

Оскільки *Web* з'явилася як документо-орієнтована мережа, питання організації електронного документообігу в мережах *Internet* (*Intranet*) вирішені достатньо прозоро як для базових мережевих сервісів, так і для відповідних протоколів.

Протягом останніх декількох років можна спостерігати якісні зміни, які зазнає *World Wide Web*. Від сукупності серверів, що містять статичні документи з посиланнями один на одного, сучасний *Web* практично неможливо уявити без інтерактивних *Web*-додатків, які обробляють і поміщають в бази даних для користувача введення, динамічно генерують сторінки на скриптових мовах за запитом користувача і, таким чином, обслуговують ту чи іншу сферу діяльності людини, використовуючи при цьому широкий спектр технологій, що реалізують подібну функціональність (*CGI, ISAPI, ASP, JSP* тощо).

Взаємна інтеграція цільових завдань різних організацій і установ, що відбувається зараз в усьому світі, неминуче тягне за собою появу технологій і стандартів інтеграції обслуговуючих їх додатків і корпоративних інформаційних систем. Найбільш популярною технологією такої інтеграції на цей час слід назвати новітню технологію *ASP.NET*, що була запропонована компанією *Microsoft*. *ASP.NET* забезпечує обмін даними у форматі *XML* (*eXtensible Markup Language*) за протоколом *SOAP* (*Simple Object Access Protocol*) і створення *Web*-сервісів, які застосовують подібний обмін даними.

Web-сервіс – це, по суті, об'єкт, який реалізує один або кілька методів, до яких можна звертатися через *Internet* (*Intranet*) з будь-якого іншого застосування і які відповідно до технології *ASP.NET* базуються на трьох основних *Web*-стандартах: *SOAP*-протокол, що забезпечує посилку повідомлень за протоколом *HTTP*; *WSDL* (*Web Services Description Language*) – мова для опису програмних інтерфейсів *Web*-сервісів; *UDDI* (*Universal Description, Discovery and Integration*) – стандарт для індексації *Web*-сервісів.

Сервери додатків у трирівневих архітектурах „клієнт-сервер" є сховищами *Web*-сервісів і роблять їх доступними через протоколи *HTTP GET, HTTP POST* і *HTTP SOAP*.

Існуючі *Web*-сервіси описуються в *WSDL*-документах, що розташовуються або на сервері-додатків, або в спеціальних *XML*-сховищах. *WSDL*-документ може посилатися на інші *WSDL*-документи і документи *XSD* (*XML Schema*), в яких описано типи даних, що використовуються *Web*-сервісами. *XML*-сховища використовуються для керування *WSDL*-документами. Усередині *WSDL*-документа знаходиться адреса (*URL*) *Web*-сервісу.

Web-сервіс можна вважати механізмом, що передає логіку додатка іншим Web-додаткам, які по суті є клієнтами по відношенню до сервісу. Web-сервіс може бути визначений, як компонент багаторазового використання, що може застосовуватись у різних Web-додатках. Головна привабливість сервісів полягає у тому, що вони виконуються незалежно від мови або об'єктної моделі, що використовується.

Трирівневу архітектуру „клієнт-сервер" на основі WEB-сервісу подано на рис. 1.

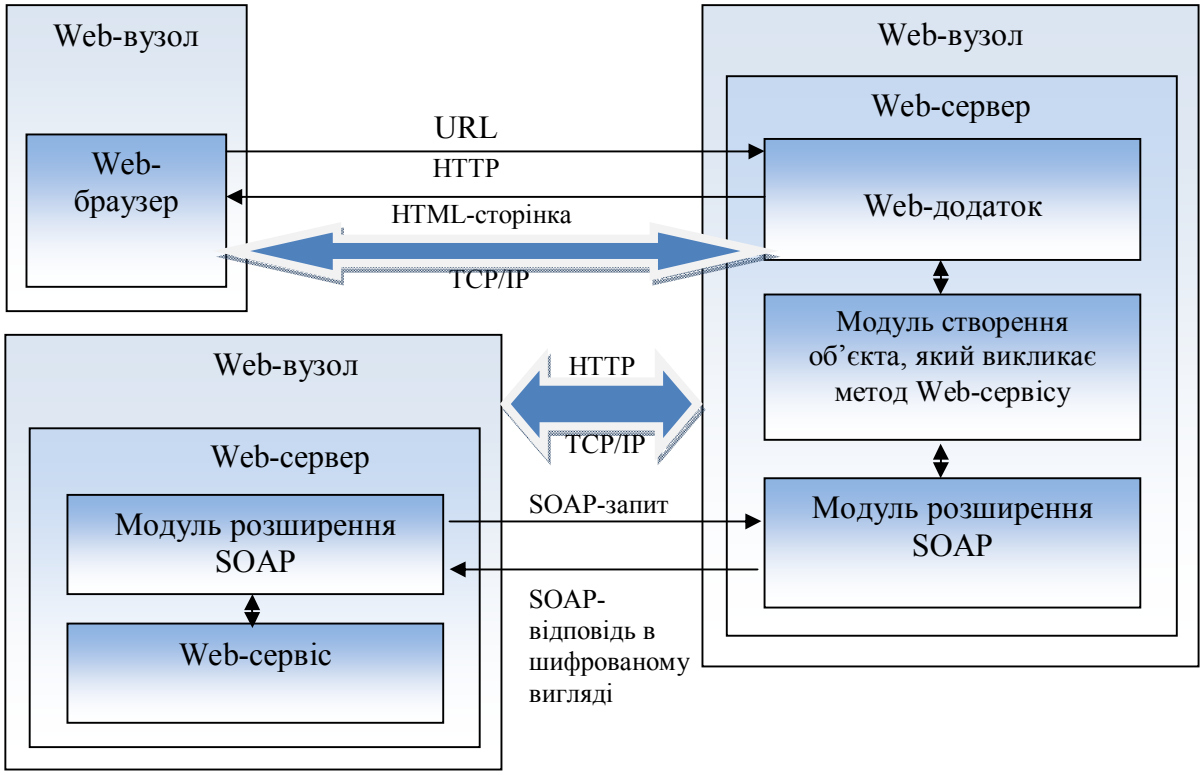

Рис. 1. Трирівнева архітектура „клієнт-сервер" на основі *WEB*-сервісу

Web-клієнт та *Web*-сервіс взаємодіють на апаратному рівні за протоколом *TCP/IP*, та на логічному рівні – за протоколом *HTTP* з укладеними в нього принципами *SOAP*.

Як показано на схемі функціонування (рис. 1), користувач на своєму комп'ютері за допомогою *Web*-браузера, після введенням *URL*-адреси звертається до *Web*-форми на *Web*-сервері, де розміщений *Web*-додаток. За допомогою модуля створення об'єкта створюється об'єкт, який формує *SOAP*-запит, а після цього отримує *SOAP*-відповідь від

Web-сервісу. Модуль розширення *SOAP* на *Web*-серверах призначено для інтерпретації *SOAP*-повідомлень. Процес, що відбувається при виклику Web-сервісу, аналогічний звичайному виклику методу. Основна відмінність полягає в тому, що замість виклику методу, створюється об'єктом клієнту *SOAP*-повідомлення (*SOAP*-запит), через яке за вказаним транспортним протоколом клієнт звертається до об'єкта *Web*-сервісу, а саме методу. Після отримання *SOAP*-запиту до методу *Web*-сервісу він обробляє його і відправляє результат мережею назад об'єкту *Web*-клієнта у вигляді *SOAP*-повідомлення – *SOAP*-відповідь.

Для організації захищеного обміну *SOAP*-повідомленнями, а саме *SOAP*-відповіді, необхідно інтегрувати в модулі інтерпретації інформаційного *SOAP*-повідомлення методи криптографічного захисту даних.

Відповідно до схеми (рис. 2) платформа *.NET Framework* серіалізує і десеріалізує *XML*-структуру на обох етапах обробки: як на *Web*-сервері *Web*-сервісу, так і на *Web*-сервері, де розміщений *Web*-додаток. У цю інфраструктуру можна додати розширення *SOAP* для перевірки або зміни *SOAP*-повідомлень до і після кожного з цих етапів серіалізації і десеріалізації.

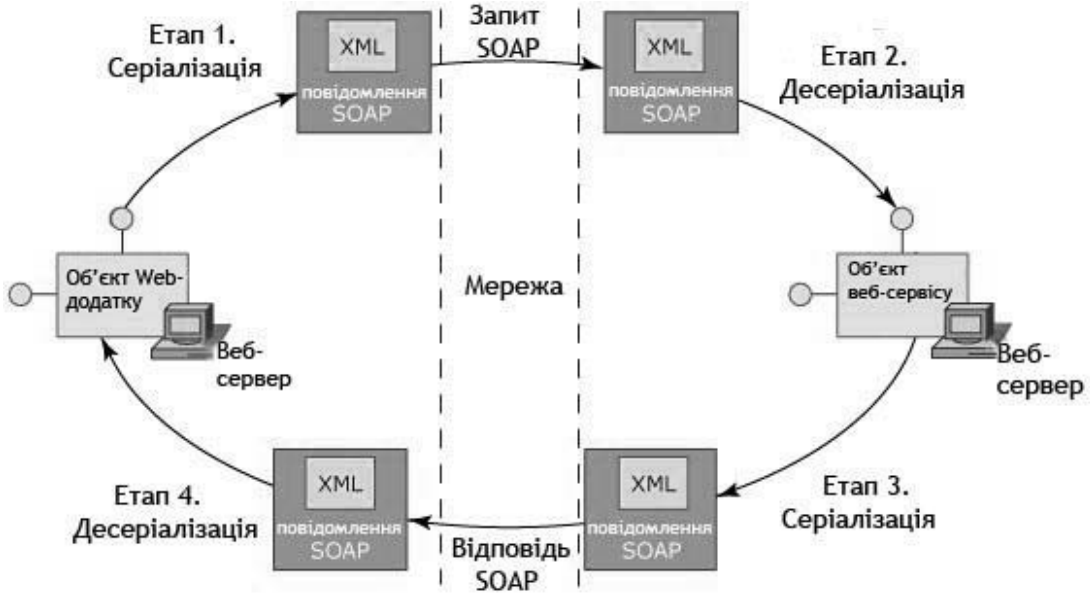

Рис. 2. Процес взаємодії між Web-додатком і Web-сервісом

Наприклад, за допомогою розширення *SOAP* можливо реалізувати шифрування, яке може зашифрувати частину *SOAP*-повідомлення, після того, як платформа *.NET Framework* серіалізує аргументи *Web*-сервісу, а потім розшифрувати *SOAP*-повідомлення безпосередньо на *Web*-сервері *Web*-додатку, перш ніж *.NET Framework* десеріалізує *SOAP*-відповідь. Як правило, якщо розширення *SOAP* змінює вміст *SOAP*-повідомлення, то зміни повинні бути виконані на обох сторонах інформаційного обміну.

На рис. 3 представлена робота *Web*-додатку з реалізованими розширеннями *SOAP*.

Викликаючий об'єкт *Web*-додатку формує *SOAP*-запит до *Web*-сервісу, після *XML*-серіалізації (*AfterSerialize*) блок копіювання потоку копіює потік додатку в потік мережі, який в свою чергу за протоколом *SOAP* передається на *Web*-сервіс.

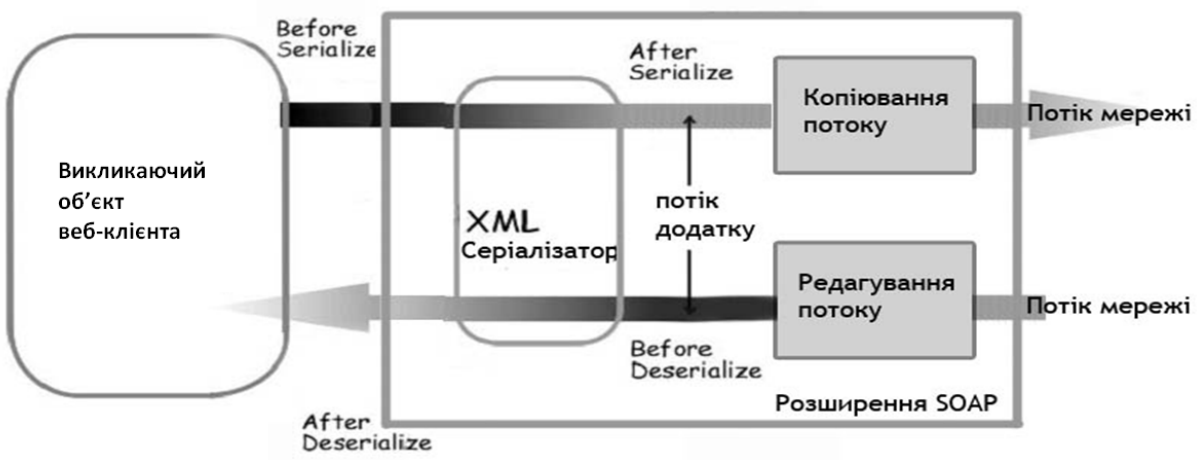

Рис. 3. Розширення *SOAP* на *Web*-додатку

Потік мережі, який відправляє *Web*-сервіс, потрапляє до блоку редагування потоку *Web*-додатку, де за допомогою розширень *SOAP* відбувається розшифрування *SOAP*-відповіді та копіювання в потік додатку на етапі до серіалізації (*BeforeSerialize*). *XML*-серіалізатор в свою чергу після етапу серіалізації повертає дані, які сформував метод *Web*-сервісу, до викликаючого об'єкта *Web*-клієнта.

На рис. 4 показано роботу *Web*-сервісу з реалізованими розширеннями *SOAP*. На етапі до серіалізації (*BeforeDeserialize*) потік мережі копіюється в потік додатку за допомогою блоку копіювання потоку, після чого потік додатку проходить етап *XML*-серіалізації до методу *Web*-сервісу, далі *SOAP*-запит від веб-клієнта обробляється та сформована *SOAP*-відповідь від Web-сервісу після етапу *XML*-серіалізації (AfterSerialize), вже в представленні потоку додатку проходить блок редагування потоку, а саме шифрування *SOAP*-повідомлення та копіюється в потік мережі, який безпосередньо за протоколом *SOAP* передається на *Web*-додаток, який звернувся до методу *Web*-сервісу.

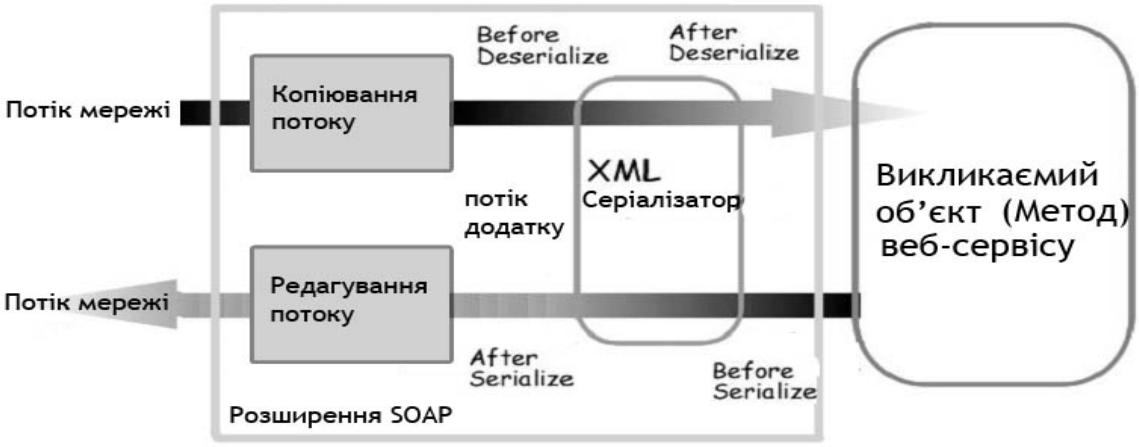

Рис. 4. Розширення SOAP на Web-сервісі

Для організації захищеного інформаційного обміну *SOAP*-повідомленнями між *Web*-сервісом та *Web*-додатком пропонується скористатися математичною моделлю на основі симетричного алгоритму *AES*.

Шифр *AES* (*Rijndael*) – це симетричний блоковий шифр, що оперує блоками даних розміром 128 біт і довжиною ключа 128, 192 або 256 біт.

В якості вихідних даних для методу шифрування виступає *SOAP*-повідомлення, що містить важливу інформацію та передається в мережах спеціального призначення, ключ для шифрування та дешифрування, вектор ініціалізації.

Цільовою настановою удосконалення методу шифрування *SOAP*-повідомлення за алгоритмом *AES* є необхідність підвищення показника шифрозахищеності даних шляхом застосування операції шифрування сегмента корисних даних структури *SOAP*-повідомлення, що містить важливу інформацію і передається в мережах спеціального призначення.

Перед початком застосування алгоритму симетричного блокового шифрування необхідно виділити сегмент корисних даних із структури *SOAP*-повідомлення, який надалі буде зашифрований для забезпечення безпеки передачі в мережі спеціального призначення.

Для визначення формату блоків даних і числа раундів шифрування необхідно ввести наступні поняття:

Nb – число 32-бітних слів, що містяться у вхідному блоці, *Nb* = 4;

Nk – число 32-бітних слів, що містяться в ключі шифрування, *Nk* = 4,6,8;

Nr – кількість раундів шифрування, як функція від *Nb* і *Nk*, *Nr* = 10,12,14.

Вхідні (*input*), проміжні (*state*) і вихідні (*output*) результати перетворень, які виконуються в рамках алгоритму, називаються станами (State). Стан можна представити у вигляді матриці 4 x *Nb*, елементами якої є чотири рядки по *Nb* байт в порядку *S*00, *S*10, *S*20, *S*30, *S*01, *S*11, *S*21, *S*31 тощо.

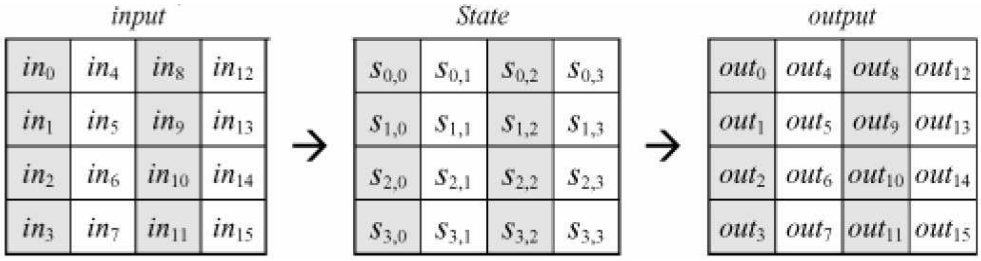

Рис. 5. Архітектура „Квадрат”

На старті процесів шифрування і дешифрування масив вхідних даних *in*0, *in*1, ... , *in*15 перетворюється в масив *State* за правилом (1):

$$s[r,c] = in[r + 4c], \quad (1)$$

де $0<r< 4$ и $0<c<Nb$. Наприкінці дії алгоритму виконується зворотне перетворення (2):

$$out[r + 4c] = s[r, c], \quad (2)$$

де $0<r<4$ и $0<c<Nb$. Вихідні дані виходять з байтів стану у тому ж порядку.

Чотири байти в кожному стовпці стану являють собою 32-бітне слово, якщо *r* – номер рядка в стані, то одночасно він є індексом кожного байта в цьому слові. Отже, стан можна представити як одномірний масив 32-бітових слів *w*0, ... , *wNb*, де номер стовпця стану є індекс у цьому масиві.

Ключ шифрування також як і масив *State* представляється у вигляді прямокутного масиву з чотирма рядками. Кількість стовпців цього масиву дорівнює *Nk*.

Для алгоритму *AES* число раундів *Nr* визначається на старті залежно від значення *Nk*.

	Nk	*Nb*	*Nr*
AES – 128	4	4	10
AES – 192	6	4	12
AES - 256	8	4	14

Розглянемо функцію зашифрування даних, для чого введемо такі позначення:

– *SubBytes*: заміна байтів – побайтова нелінійна підстановка в *State*-блоках (*S-Box*) з використанням фіксованої таблиці замін розмірністю 8x256;

– *ShiftRows*: зсув рядків – циклічний зсув рядків масиву *State* на різну кількість байт;

– *MixColumns*: перемішування стовпців – множення стовпців стану, що розглядаються як многочлени над *GF*(28);

– *AddRoundKey*: складання з раундовим ключем – порозрядному *XOR* вмісту *State* з поточним.

Після заповнення масиву *State* елементами вхідних даних до нього застосовується перетворення *AddRoundKey*, далі, залежно від величини *Nk* масив *State* піддається трансформації раундової 10, 12 або 14 разів, причому фінальний раунд є дещо укороченим – в ньому відсутнє перетворення *MixColumns*. Вихідними даними описаної послідовності операцій є шифротекст – результат дії функції зашифрування *AES*.

Розглянемо функцію зворотного розшифрування. Якщо замість *SubBytes*, ShiftRows, *MixColumns* і *AddRoundKey* у зворотній послідовності виконати інверсні їм перетворення, можна побудувати функцію зворотного розшифрування. При цьому порядок використання раундових ключів є зворотним по відношенню до того, який використовується при зашифровуванні.

Відповідно до алгоритму генерації ключів потрібно ввести такі позначення:

Rcon[] – масив 32-бітових раундових констант;

RotWord – операція циклічної перестановки вхідного 4-байтного слова у вихідну за наступним правилом [*a*0, *a*1, *a*2, *a*3] ->[*a*1, *a*2, *a*3, *a*0];

SubWord – операція заміни в 4-байтному слові за допомогою *S-Box* кожного байта;

□ – операція виключального АБО (*XOR*).

Раундові ключі виходять з ключа шифрування за допомогою алгоритму генерації ключів. Він містить два компоненти: розширення ключа (*Key Expansion*) та раундовий ключ (*Round Key Selection*) (рис. 6). Базові елементи алгоритму виглядають наступним чином:

– загальне число бітів раундових ключів дорівнює довжині блоку, помноженої на кількість раундів, плюс 1;

– ключ шифрування розширюється в розширений ключ (*ExpandedKey*);

– раундові ключі беруться з розширеного ключа наступним чином: перший ключ містить перші *Nb* слів, другий – наступні *Nb* слів тощо.

Розширений ключ являє собою лінійний масив *w*[*i*], що складається з чотирьох (*Nr*+1) 4-байтових слів, $i = 0,4(Nr+1)$.

Перші *Nk* слів містять ключ шифрування. Кожне наступне слово *w*[*i*] виходить за допомогою *XOR* попереднього слова *w*[*i*-1] і слова на *Nk* позицій раніше *w*[*i* - *Nk*]:

$$w[i] = w[i-1] \square w[i - Nk]. \quad (3)$$

Рис. 6. Процедура розширення та вибірки раундового ключа

Для слів, позиція яких кратна *Nk*, перед *XOR* застосовується перетворення до *w*[*i*-1], а потім ще додається раундова константа *Rcon*[*i*]. Перетворення реалізується за допомогою двох додаткових операцій: *RotWord* и *SubWord*.

Значення *Rcon*[*j*] дорівнює 2*j*-1. Значення *w*[*i*] в цьому випадку визначається виразом:

$$w[i] = SubWord(RotWord(w[i-1])) \square Rcon[i / Nk] \square w[i - Nk], \quad (4)$$

де *i*-тий раундовий ключ вибирається з слів масиву розширеного ключа в проміжку від *W*[*Nb*· *i*] до *W*[*Nb*·(*i* +1)].

Для функції зворотного розшифрування використовується цей самий ключ, але в зворотній послідовності, починаючи з останнього раундового підключа зашифрування. Раундове перетворення складається з послідовного застосування до масиву State ряду трансформацій: заміна байт (*SubBytes*); зсув рядків (*ShiftRows*); перемішування стовпців (*MixColumns*); додавання раундового ключа (*AddRoundKey*).

У функціях зашифрування та розшифрування виробляються такі перетворення заміни байтів:

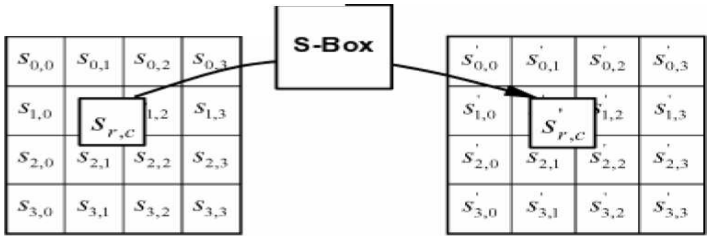

Рис. 7. Перетворення стану за допомогою *S-Box*

Використання попередньо обчисленої таблиці заміни *S-Box* зводить операцію *SubBytes* до найпростішої вибірки байтів з масиву:

$$\lambda(f) = S\text{-}box[f] \quad (5)$$

Логіка роботи *S-Box* при перетворенні байту {*xy*} відображена на рис. 8:

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рис. 8. Таблиця *S-Box* заміни байт

У функціях розшифрування застосовується операція, зворотна *SubBytesmaInvSubBytes*, яка реалізується так само просто, як і попередня за допомогою інверсної таблиці *S-Box*: $\lambda(f) = InvSbox[f]$. Її логіка роботи при перетворенні байту {xy} відображена на рис. 9:

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Рис. 9. Таблиця *S-Box* зворотної заміни байт

У перетворенні зсуву рядків останні 3 рядки стану циклічно зсуваються вліво на різне число байтів. Значення зрушень залежать від довжини блоку *Nb*. У стандарті *AES*, де визначено єдиний розмір блоку, рівний 128 бітам, рядок 1 зсувається на 1 байт, рядок 2 – на 2 байти, і рядок 3 – на 3 байти.

Операція зсуву вмісту стану може бути представлена наступним виразом:

$$S'r,c = Sr,(c+shift(r, Nb))modNb \quad (6)$$

для $0 < r < 4$ и $0 < c < Nb$, де значення $shift(r, Nb)$ залежить від номера рядка r.

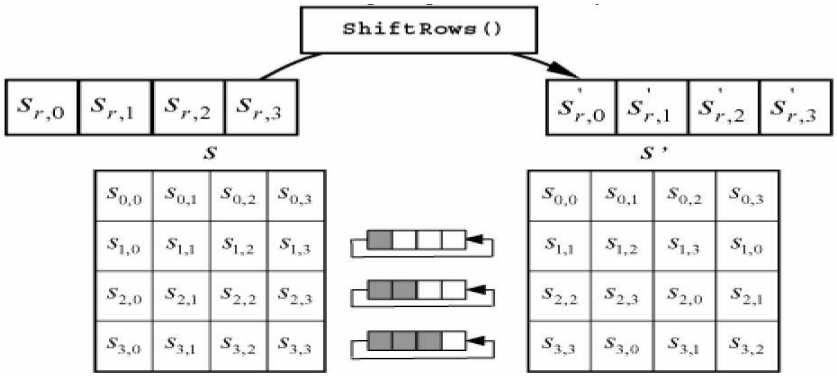

Рис. 10. Вплив перетворення *ShiftRows* на стан

У перетворенні зворотного зсуву рядків *InvShiftRows* останні 3 рядки стану циклічно зсуваються вправо на різне число байтів. Рядок 1 зсувається на 1 байт, рядок 2 – на 2 байти, і рядок 3 – на 3 байти.

Операція зсуву вмісту стану може бути представлена наступним виразом:

$$S'r,(c+shift(r,Nb))modNb = Sr,c \quad (7)$$

для $0 < r < 4$ і $0 < c < Nb$, де значення $shift(r, Nb)$ залежить від номера рядка r.

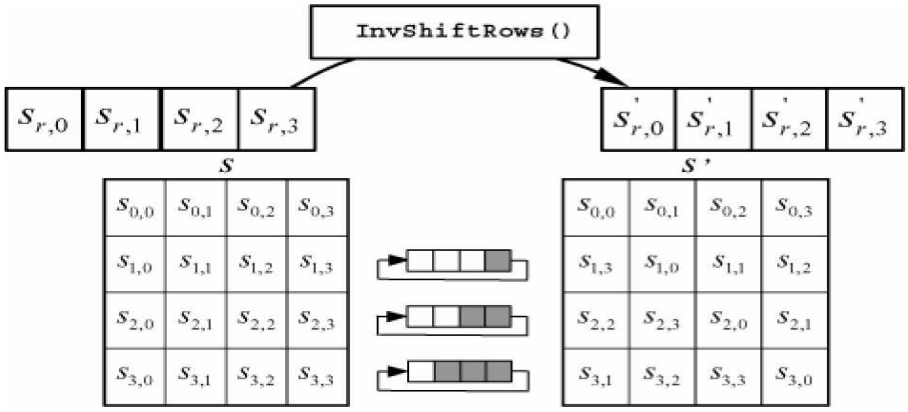

Рис. 11. Перетворення зворотного зсуву рядків

У перетворенні перемішування стовпців стовпці стану розглядаються як многочлени над *GF*(28) та множаться по модулю *x*4+ 1 на многочлен c(x), що виглядає, як:

$$c(x) = \{03\}\ x3 + \{01\}x2 + \{01\}\ x+ \{02\}. \quad (8)$$

Це перетворення може бути представлене у матричному вигляді наступним чином:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, \ 0 \le c \le 3, \quad (9)$$

де *c* – номер стовпця масиву *State*.

У результаті такого множення байти стовпця *s*0,c, s1,c, s2,c, s3,c замінюються відповідно на байти:

$$\begin{aligned} s'_{0c} &= (\{02\} \bullet s_{0c}) \oplus (\{03\} \bullet s_{1c}) \oplus s_{2c} \oplus s_{3c}, \\ s'_{1c} &= s_{0c} \oplus (\{02\} \bullet s_{1c}) \oplus (\{03\} \bullet s_{2c}) \oplus s_{3c}, \\ s'_{2c} &= s_{0c} \oplus s_{1c} \oplus (\{02\} \bullet s_{2c}) \oplus (\{03\} \bullet s_{3c}), \\ s'_{3c} &= (\{03\} \bullet s_{0c}) \oplus s_{1c} \oplus s_{2c} \oplus (\{02\} \bullet s_{3c}). \end{aligned} \quad (10)$$

У зворотному перетворенні стовпці стану розглядаються як многочлени над *GF*(28), але, звичайно, піддаються зворотному перетворенню, тобто множаться по модулю *x*4+ 1 на многочлен *d*(*x*), що виглядає наступним чином:

$$d(x) = \{0b\}\ x3 + \{0d\}\ x2 + \{09\}x + \{0e\}. \quad (11)$$

Це може бути представлене у матричному вигляді наступним чином:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, \ 0 \le c \le 3, \quad (12)$$

де *c* – номер стовпця масиву *State*.

У результаті виходять наступні байти:

$$
\begin{aligned}
s'_{0c} &= (\{0e\} \bullet s_{0c}) \oplus (\{0b\} \bullet s_{1c}) \oplus (\{0d\} \bullet s_{2c}) \oplus (\{09\} \bullet s_{3c}), \\
s'_{1c} &= (\{09\} \bullet s_{0c}) \oplus (\{0e\} \bullet s_{1c}) \oplus (\{0b\} \bullet s_{2c}) \oplus (\{0d\} \bullet s_{3c}), \\
s'_{2c} &= (\{0d\} \bullet s_{0c}) \oplus (\{09\} \bullet s_{1c}) \oplus (\{0e\} \bullet s_{2c}) \oplus (\{0b\} \bullet s_{3c}), \\
s'_{3c} &= (\{0b\} \bullet s_{0c}) \oplus (\{0d\} \bullet s_{1c}) \oplus (\{09\} \bullet s_{2c}) \oplus (\{0e\} \bullet s_{3c}).
\end{aligned} \qquad (13)
$$

У цій операції раундовий ключ додається до стану за допомогою порозрядного *XOR*. Довжина ключа (в 32-розрядних словах) дорівнює довжині блоку *Nb*.

Перетворення, що містить додавання раундового ключа до стану представляється виразом:

$$[s'0, c, s'1, c, s'2, c, s'3, c] = [s0, c, s1, c, s2, c, s3, c] \square [wr(Nb+c)], \qquad (14)$$

де $0 < c < Nb$ і $0 < r < Nr$.

Отже, можна сформулювати основні особливості стандарту *AES*, що обумовлюють його застосування в якості технологічного рішення з шифрування відкритого трафіка між *WEB*-клієнтом і *WEB*-сервісом для мереж спеціального призначення:

– запропонована архітектура „Квадрат” забезпечує швидке розсіювання і перемішування інформації, при цьому за один раунд перетворенню піддається весь вхідний блок SOAP-повідомлення, яке повинно бути конфіденційним;

– усі раундові перетворення операцій в кінцевих полях дозволяють ефективну апаратну і програмну реалізацію на різних платформах.

Запропоноване рішення спрямовано не тільки на вирішення завдання шифрування відповідного відкритого трафіка між сторонами інформаційного обміну мережі спеціального призначення, але й на скорочення показника часу виконання операцій криптозахисту *SOAP*-повідомлення за алгоритмом блокового симетричного шифрування *AES* шляхом застосування криптозахисту до сегментів корисних даних структури *SOAP*-повідомлення.

Література:

1. *Зензин О.С.* Стандарт криптографической защиты – AES. Конечные поля / О.С. Зензин, М.А. Иванов. – М. : КУДИЦ-ОБРАЗ, 2002. – 176 с.

2. *Киви Берд* Конкурс на новый криптостандарт AES / Киви Берд. // Системы безопасности связи и телекоммуникаций. – 1999. – № 27-28.

3. *Мэтью Мак-Дональд* Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов. : Пер. с англ. / Мэтью Мак-Дональд, Марио Шпушта. – М. : ООО „И.Д. Вильямс”, 2006. – 1351 с.

4. *Ричард Андерсон* ASP.NET для профессионалов / Андерсон Р., Фрэнсис Б., Хомер А. и др. – Санкт-Петербург : Лори, 2004. – 1260 с.

5. *Роб Камерон* ASP.NET 3.5, компоненты AJAX и серверные элементы управления для профессионалов / Роб Камерон, Дэйл Михалк. – Санкт-Петербург : Вильямс, 2009. – 608 с.

6. *Столлингс Вильям* Криптография и защита сетей: принципы и практика / Столлингс Вильям.– М. : ООО „И.Д. Вильямс”, 2001. – 672 с.

7. *Шнайер Брюс* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке #C / *Шнайер Брюс* – М. : Издательство „ТРИУМФ”, 2002. – 595 с.