УДК 004.75

І.Ф. АУЛОВ, Ю.І. ГОРБЕНКО, канд. техн. наук

РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНОЇ БІБЛІОТЕКИ ДЛЯ ЗАХИСТУ КІНЦЕВИХ КОРИСТУВАЧІВ В СЕРЕДОВИЩІ ХМАРИ ДЛЯ МОДЕЛІ SAAS

Вступ

Поширення різноманітних ОС, веб-технологій та технологій хмарних обчислень, мобільних пристроїв з доступом в Інтернет сприяло зміні моделі до розповсюдження та надання доступу до програмного забезпечення користувачам. Розробники програмного забезпечення переорієнтувалися з програм для персональних комп'ютерів та окремих ОС, до більш гнучкої моделі, де в якості середовища виконання програми виступає веб-додаток, а користувач має можливість доступу до програми та своїх даних через веб-браузер з будь-якого пристрою. Для запобігання несанкціонованого доступу до облікового запису користувача та його даних, гарантування конфіденційності та цілісності даних користувача, що зберігаються на стороні постачальника програмного забезпечення, надання послуг ЕЦП необхідним компонентом програмного забезпечення є криптографічна бібліотека або модуль, що буде працювати на стороні клієнта та надавати доступ до основних функцій за допомогою мови програмування JavaScript.

1. Огляд існуючих технологій

Основними технологіями, за допомогою яких можна реалізувати криптографічну бібліотеку для надання послуг захисту інформації кінцевим користувачам в веб-застосунках є: Oracle Java Applet, Microsoft Active X, Netscape API (NAPI), Google Native Client (NaCl) та Google Portable Native Client (PNaCl), JavaScript. Розглянемо детально кожну з цих технологій.

1.1. Технологія додатків Java Applet

Java Applet – це прикладна програма, що виконується в веб-браузері з використанням віртуальної машини Java (JVM). Технологія Java апплетів з'явилася з першої версії мови програмування Java в 1995 році та використовується для надання більш широких можливостей веб-застосунку, що не можуть бути надані стандартними засобами веб-браузера. Java апплет виконується за допомогою додатково встановлених плагінів веб-браузера, що надаються Oracle безкоштовно. Більш широкі можливості Java апплетів, в тому числі виконання нативного коду, породжують проблеми безпеки. Для вирішення цих проблем компанія Oracle використовує технологію запуску додатків в «пісочниці» та надання апплету дозволів від користувача. Окрім цього, Oracle постійно виправляє помилки та вдосконалює безпеку технології та випускає оновлення, що створюють певні проблеми для розробників додатків, оскільки при новому оновленні стара версія JVM блокується, в тому числі й засобами браузера [1].

Для написання Java апплетів використовується мова програмування Java, що, по-перше, дозволяє реалізовувати кроссплатформенні додатки, а по-друге - за допомогою технології нативних інтерфейсів Java (JNI) використовувати існуючі бібліотеки, написані мовою C/C++. Використання JNI дозволяє скоротити час розробки додатка, підвищити його швидкодію, а також надає можливість доступу до системних функцій та інтерфейсу пристроїв, відсутніх в Java. Безумовним недоліком цієї технології є необхідність підтримки бібліотек окремо для кожної з платформ. Вбудований в веб-сторінку аплет може самостійно надавати графічний інтерфейс для взаємодії з користувачем, або за допомогою технології LiveConnect (зв'язку JavaScript з Java) надавати безпосередній доступ до функцій апплета.

1.2. Технологія керуючих елементів Active X

ActiveX – програмний фреймворк, що розроблений Microsoft в 1996 році як розвиток технології Component Object Model (COM) та Object Linking and Embedding (OLE) та використовується в ОС Windows. Керуючі елементи ActiveX можуть бути використані як окремий компонент в програмах або як додаток до веб-браузера, аналогічно до технології Java Applet, але на відміну від неї офіційно підтримується тільки браузером Microsoft Internet Explorer та ОС Windows. Створення керуючих елементів ActiveX можливе за допомогою мов програмування, що підтримують розробку COM об'єктів, таких, наприклад, як C++, C#, Borland Delphi. Аналогічно до Java аплетів керуючі елементи ActiveX підтримують завантаження та використання сторонніх нативних бібліотек. Виклик функцій ActiveX елементу на сторінці веб-браузера виконується з використанням мови JavaScript [2].

Можливість автоматичного завантаження керуючих елементів ActiveX з веб-сторінки, а також їх запуск та виконання з правами процесу, що їх запустив, призводять до проблем безпеки та роблять їх одним з векторів атак зловмисників. Основним методом захисту від виконання зловмисного програмного забезпечення, що використовує технологію керуючих елементів ActiveX, є управління правами встановлення та запуску додатків в браузері. Управління правами дозволяє достатньо гнучко встановлювати налаштування та надавати права лише додаткам надійних розробників.

1.3. Технологія додатків NPAPI

Програмний інтерфейс модулів Netscape, що підключаються (англ. Netscape Plugin Application Programming Interface, NPAPI), – кроссплатформенна архітектура плагінів, що була розроблена в 1995 році для веб-браузера Netscape. На сьогодні цю технологію підтримують настільні веб-браузери, окрім Microsoft Internet Explorer та Google Chrome (відмовиться від підтримки в вересні 2015).

Розробка модуля виконується з використанням мови програмування C/C++ та включає в себе реалізацію сталого набору функцій інтерфейсу необхідних для завантаження, ініціалізації та роботи модуля. Робота модуля в різних ОС потребує окремої збірки для кожної з ОС, що при використанні специфічних системних функцій є недоліком. NPAPI плагіни підтримують завантаження та використання нативних бібліотек, що встановлені в системі, а також взаємодію з веб-браузером з використанням мови програмування JavaScript [3].

На відміну від керуючих елементів ActiveX, завантаження яких можливо автоматично з веб-сайту за згоди користувача, плагіни NPAPI потребують інсталяції користувачем з сайту розробника або з довіреної точки розповсюдження. Таким чином, безпека плагінів NPAPI забезпечуються лише за рахунок відсутності можливості автоматичного їх встановлення браузером з веб-сайту який його потребує для роботи.

1.4. Технологія додатків NaCl та PNaCl

Технологія додатків Native Client (NaCl) представляє собою середовище пісочниці, яке дозволяє запускати скомпільований C/C++ код в браузері ефективно і безпечно, автоматично завантажуючи необхідні для роботи модулі в залежності від архітектури ОС користувача. В результаті розвитку технології NaCl з'явилася технологія портативного Native Client (Portable Native Client, PNaCl), що дозволяє розробнику компілювати C/C++ код незалежно від операційної системи та архітектури платформи, де він буде виконуватися. Хоча технологія є відкритою, але сьогодні додатки NaCl та PNaCl підтримують лише браузери Google Chrome та Opera [4]. Доступ до функцій модуля відбувається за рахунок використання мови JavaScript.

Головною перевагою використання цієї технології є безпека виконання коду, швидкість якого наближена до швидкості роботи нативного коду. Середовище пісочниці реалізується за рахунок використання своєї реалізації системної бібліотеки Newlib або її альтернативного варіанту GNU glibc. У зв'язку з використанням власної реалізації системної бібліотеки та неможливістю звертатися до системних бібліотек, суттєвим недоліком технології є неможли-

вість використання апаратних засобів, в тому числі пристроїв USB, що суттєво звужує використання технології для захисту інформації.

1.5. Використання мови програмування JavaScript

Мова програмування JavaScript є розширенням мови ECMAScript, що стандартизована міжнародною організацією ECMA в специфікації ECMA-262 та підтримується більшістю браузерів [5].

Для вирішення задачі підвищення швидкодії з обробки великих об'ємів даних на стороні клієнта в середовищі браузера було запропоновано внести до стандарту JavaScript об'єкти з строгою типізацією та функції роботи з ними. Також Mozilla було запропоновано використання типізованої підмножини JavaScript – asm.js [6], яка дозволяє визначати типи змінних на етапі компіляції та оптимізувати виконання коду інтерпретатора в браузері, що підтримує asm.js. При цьому було збережено зворотну сумісність з існуючим стандартом мови JavaScript, що дозволило виконувати код, написаний з використанням asm.js в браузерах, які його не підтримують.

Поява типізованих об'єктів, зокрема масивів, та використання підмножини asm.js, створити компілятор Emscripten [7], який дозволяє достатньо швидку реалізувати криптографічну бібліотеку або додаток, використовуючи існуючий код на мові програмування C/C++.

Іншим напрямком підвищення швидкодії веб-додатків написаних з використанням мови JavaScript є розробка спеціалізованих інтерфейсів JavaScript, що реалізуються веб-браузерами та дозволяють значно підвищити виконання окремих функцій, наприклад криптографічних. Розробкою стандартів в цьому напрямі займається міжнародне співтовариство World Wide Web Consortium (W3C), до складу якого входять велика кількість організацій [8]. На сьогодні основним стандартом, що розробляється в галузі захисту криптографічної інформації, є Web Cryptography API. Цей стандарт надає стандартний інтерфейс управління ключами, доступу до функцій гешування, підпису, шифрування. Головними недоліками цього стандарту в першу чергу є необхідність його впровадження розробниками веб-браузерів, що потребує часу, несе ризики несумісності реалізацій та неможливість розширення набору стандартних криптографічних перетворень, що є актуальним для його використання в України.

Таким чином, використання мови JavaScript для реалізації веб-застосунків вирішує задачі інтероперабельності та роботи на різних платформах, а також виконання коду на стороні користувача в безпечному середовищі веб-браузера. Головним недоліком використання мови JavaScript, як і у випадку з додатками NaCl та PNaCl, є відсутність механізмів взаємодії з апаратними засобами, не використовуючи додатки до веб-браузерів або java-апплети.

2. Порівняння технологій реалізації криптографічних бібліотек

2.1. Вимоги до криптографічної бібліотеки

Аналіз послуг, що повинні надаватися криптографічною бібліотекою та середовища, в якому вона використовується, дозволяє сформулювати наступний перелік вимог до криптографічної бібліотеки, що використовується кінцевими користувачами в веб-браузері:

кроссплатформеність – можливість бібліотеки працювати на різних платформах та різних браузерах;

швидкодія – бібліотека повинна виконувати операції за прийнятний час;

об'єм оперативної пам'яті, що споживається під час роботи, та розмір бібліотеки;

автоматичне завантаження та інсталяція бібліотеки;

управління завантаженням, правами та рівнем доступу бібліотеки;

механізми захисту коду бібліотеки від модифікації та аналізу;

доступ до апаратних криптографічних засобів та носіїв ключа;

простота використання інтерфейсу бібліотеки розробником веб-додатку;

доступ до файлової системи користувача;

взаємодія з серверами в Інтернеті.

2.2. Порівняння технологій реалізації криптографічної бібліотеки за критерієм кроссплатформеності та механізмом захисту

Використовуючи вимоги, що зазначені в п. 2.1, було сформовано критерії та проведено порівняння технологій реалізації криптографічної бібліотеки (табл. 1).

Таблиця 1

Критерій	Технологія реалізації криптографічної бібліотеки				
	Java Applet	Active X	NACL	NAPI	JavaScript (asm.js + Emscripten)
1. Архітектури процесорів, що підтримуються	x86-32, x86-64, IA64, SPARK, ARM, PPC32/64	x86-32, x86-64, IA64,	x86-32, x86-64, ARM	x86-32, x86-64, IA64, SPARK, ARM, PPC32/64	Всі архітектури, що мають веб-браузер з JavaScript
2. ОС, що підтримуються	Windows, Mac OS, Linux, Unix, Sun Solaris, AIX	Windows	Windows, Mac OS, Linux, Chrome OS	Windows, Mac OS, Linux, Unix, Sun Solaris, AIX	Всі ОС, що мають веб-браузер з JavaScript
3. Браузери, що підтримують технологію	Internet Explorer, Mozilla Firefox, Safari (Mac OS x64), Google Chrome (до 2015 р.), Opera	Internet Explorer	Google Chrome (тільки для ПК)	Internet Explorer, Mozilla Firefox, Safari (Mac OS), Google Chrome (до 2015 р.), Opera	Всі браузери, що підтримують JavaScript
4. Управління безпекою додатків	Використання технології «пісочниці» та необхідність отримання прав для виконання привілейованих операцій. Перевірка видавця за допомогою ЕЦП		Використання технології «пісочниці» та необхідність отримання прав для виконання привілейованих операцій		Вимоги безпеки веб-браузера до виконання JavaScript
5. Доступ до апаратних криптомодулів та носіїв ключів	Стандартні інтерфейси (PKCS#11, SmartCard API і т.і.), через JNI	Є	Відстуній	Є	Відстуній
6. Захист коду бібліотеки від модифікації та аналізу	Перевірка цілісності за допомогою ЕЦП. Необхідно проводити обфускацію коду java	Перевірка цілісності за допомогою ЕЦП. Аналіз потребує діссассемблювання коду	Захист від модифікації відсутній. Аналіз потребує діссассемблювання коду		Захист від модифікації відсутній. Автоматичнаобфускація коду при компіляції
7. Автоматичне завантаження та інсталяція	Так (потребує встановлення JRE)	Так	Так	Так	Так
8. Доступ до файлової системи	Повний або обмежений в режимі «пісочниці»	Так	Тільки до власної файлової системи в «пісочниці» або за рахунок інтерфейсу браузера	Так	Тільки за рахунок інтерфейсу браузера

2.3. Переваги та недоліки технологій реалізації криптографічної бібліотеки

Використовуючи огляд технологій, а також результати порівняння з табл. 1, можна зробити наступні висновки щодо переваг та недоліків різних технологій (табл. 2)

Таблиця 2

Технологія	Переваги	Недоліки
Java Applet	– кросплатформеність та підтримка більшістю настільних браузерів; – підтримує кешування в більшості браузерів, що дозволяє швидко завантажуватися при поверненні на веб-сторінку; – може мати доступ до файлової системи, ресурсів ОС та апаратних модулів; – може завантажувати та використовувати нативні бібліотеки, за умови їх компіляції під кожну з платформ окремо; – захист від модифікації та контроль доступу; – незначне зниження швидкодії, якщо використовується JNI.	– вимагає установки Java-розширення (plug-in), що не у всіх браузерах доступно за замовчуванням; – вимагає актуальної JRE, в іншому випадку блокується браузером чи JRE, навіть без підключення до Інтернету; – не підтримується в мобільних браузерах, та з 2015 року не підтримується Google Chrome, та Opera; – обмежений доступ до файлової системи та апаратного забезпечення з використання технології «пісочниці»; – при використання нативних бібліотек, необхідність їх підтримки для кожної з платформ.
Active X	– може використовуватися, як COM – об'єкт при розробці інших програм, на мовах програмування, що підтримують використання COM – об'єктів (C++, C#, Delphi); – має доступ до файлової системи, ресурсів ОС та апаратних модулів; – може завантажувати та використовувати нативні бібліотеки; – захист від модифікації та контроль доступу.	– підтримка тільки в ОС Windows та браузері Internet Explorer; – автоматичне встановлення з сайту; – можливість необмеженого доступ до бібліотек ОС.
NaCl, PNaCl	– високий рівень безпеки за рахунок виконання коду в середовищі «пісочниці», що не має доступу до файлової системи користувача та до функції ОС. – простота використання - PNaCL модулі не вимагають установки; – скомпільований модуль запускається в будь-якій ОС в середовищі браузера; – незначне зниження швидкодії порівняно з реалізацією нативного модуля.	– використання власного інтерфейсу, який належить компанії Google і заснований на деталях специфічних для веб-браузера Google Chrome. З цієї причини інтерфейс не підтримується в інших веб-браузерах; – доступ до API ОС обмежений операціями, закладеними в PPAPI та бібліотеками; – помилки в реалізації стандартної бібліотеки, відсутність підтримки деяких функцій.
NPAPI	– необмежений доступ до можливостей ОС; – усталений стандарт; – простий процес установки в систему; – швидкодія на рівня нативного модуля.	– в силу необмеженості доступу до ОС представляє потенційну загрозу безпеки користувача. За статистикою, плагіни, реалізовані на NPAPI, є основним вектором атак на Web-браузери; – через проблеми з безпекою, Google анонсував, що в 2015 році NPAPI інтерфейс не буде підтримуватися в Google Chrome; – модуль необхідно підтримувати окремо для кожної з платформ.
JavaScript (asm.js + Emscripten)	– виконання коду в середовищі браузера, що підтримує JavaScript; – прискорення розробки, за рахунок компіляції існуючого коду C, C++.	– швидкодія залежить від підтримки браузером asm.js; – доступ до апаратних засобів безпеки та носіїв ключів можливий лише за рахунок встановлення додаткових розширень браузера; – обмежений доступ до файлів на файловій системі та обмеження на їх розмір.

2.4. Порівняння бібліотек за швидкодією

В якості стенду для тестування використовувався комп'ютер з ОС Windows 7 x64, процесор Intel Core i3 3,3 GHz, ОЗП 4 GB, веб-браузер Google Chrome 32 v38, JRE 8.25. Бібліотеки було скомпільовано для максимальної оптимізації за швидкодією. Для кожної з функцій, що тестувалася, було виконано 100 незалежних вимірювань часу та обрано найменший з них. Результати порівняння бібліотек представлені в табл. 3.

Таблиця 3

Показник	Мова програмування						
	C/C++	Active X	NaCl, PNaCl	NPAPI	Java	Java + JNI	JavaScript
1. Розмір бібліотеки, МБ	4,1	4,3	12,6	4,3	0,7	4,3	4,3
2. Пам'ять, МБ	16	17	21	17	4,5	21	17
3. Швидкість гешування (ГОСТ 34.311-95), Мб/с	285	250	245	243	53	234	153
4. Швидкість шифрування (ГОСТ 28147), Мб/с	465	372	394	361	220	355	272
5. Швидкість ЕЦП (ДСТУ 4145-2002, поле 257), мс	0,7/4,2	0,73/4,5	0,8/4,7	0,77/4,4	17,2/21,5	0,76/4,5	2,8/18,2
6. Швидкість НШ (ДГ - в гр.т. е.кр., поле 571), мс	6,8/6,8	7,1/7,1	7,6/7,6	7,2/7,2	50,9/51,8	7,8/7,8	21,6/21,5

3. Особливості реалізації криптографічних бібліотек за допомогою Emscripten

Emscripten представляє собою компілятор LLVM байт коду в JavaScript. LLVM байт код може бути згенеровано з файлів на мові C/C++ з використанням llvm-gcc або clang, або з будь якої іншої мови, що може бути конвертована в LLVM байт код. Сам компілятор Emscripten є програмою з відкритим кодом та поширюється з використанням двох ліцензій: MIT та ліцензією NCSA університету Іллінойса.

Розглянемо детально особливості реалізації криптографічної бібліотеки за допомогою Emscripten, реалізацію пам'яті модуля, стандартні бібліотеки, що входять до SDK, можливості роботи з файловою системою та мережею Інтернет, оптимізацію.

3.1. Реалізація криптографічного модуля

Набір бібліотек, що входе до складу SDK, реалізує основні функції стандартної бібліотеки stdlib з інтерфейсом для ОС Linux, тому платформозалежний код для ОС Linux буде компілюватися з найменшими труднощами.

Криптографічна бібліотека може бути скомпільована як завершений додаток або як модуль, що експортує функції. Підключення модуля до веб-сторінки здійснюється за допомогою включення відповідного файлу з JavaScript кодом стандартними методами веб-браузера.

Модель пам'яті скомпільованого модуля представляє собою об'єкт типу ArrayBuffer, який виділяється при ініціалізації модуля. Його розмір може бути задано трьома різними способами: під час компіляції (флаг -s TOTAL_MEMORY), перед ініціалізацією (змінна модуля Module.TOTAL_MEMORY) або змінено в процесі роботи модуля (-s ALLOW_MEMORY_GROWTH=1). Використання останньої опції не рекомендується, бо це впливає на швидкодію. Доступ до пам'яті організовано з використанням типізованих масивів: Int8Array, Int16Array, Int32Array, Uint8Array, Uint16Array, Uint32Array, Float32Array, Float64Array. Така організація пам'яті впливає на можливість доступу до неї через покажчики: приведення до покажчика типу, розмір якого більший за розмір елементу масиву, та індексація за допомогою нього зі зміщенням, не рівним розміру цього типу, призводить до помилки. Окрім цього в пам'яті модуля знаходиться стек, розмір якого визначається значенням змінної TOTAL_STACK.

Виклик функцій модуля виконується за допомогою функції модуля Module.ccall(), яка приймає в якості параметрів: ім'я функції С, тип значення, що повертається, масив типів змінних та безпосередньо масив змінних. В якості типів, що можуть бути автоматично відображені до пам'яті модуля в функції Module.ccall(), відносяться: числові типи ('number'), масиви ('array'), строки ('string'). Передача показчиків на змінні або дані довільного формату

відбувається з використанням типу 'number', попередньо в модулі виділяється пам'ять під них.

При передачі строкових параметрів до функцій модуля слід пам'ятати, що строки в JavaScript використовують кодування UTF-16, тому при роботі з строками в модулі потрібно використовувати функції, що підтримують кодування UTF-16. Також при передачі масивів їх значення копіюється в стек, тому значення TOTAL_STACK – повинно бути більшим за розмір максимально масиву, що передається.

Зважаючи на це при визначенні розміру пам'яті TOTAL_MEMORY, що необхідна модулю, обчислюється як сума максимально можливого розміру параметрів, що передаються через стек, параметрів, що використовують при передачі функції виділення, та внутрішні потреби бібліотеки – виділення пам'яті в стеку при роботі внутрішніх функцій.

3.2. Доступ до файлової системи

З міркувань безпеки доступ до файлової системи в браузері обмежено. Для роботи з файлами поза межами браузера необхідно використовувати File API [7], яке дозволяє обрати та зчитати файл з файлової системи, завантажити його до пам'яті повністю або частинами. Запис файлів безпосередньо на файлову систему користувача неможливий.

Для зберігання даних користувача в веб-браузері можна використовувати технології HTML5: Web Storage, IndexedDB або File API – інтерфейс FileWriter. Однак Web Storage станом на сьогодні підтримує тільки запис строкових даних, IndexedDB – підтримує лише асинхронну роботу та не підтримується всіма браузерами, інтерфейс FileWriter – взагалі підтримує лише Google Chrome, а подальша робота над ним призупинена. Єдиною можливістю зберігання файлів на файловій системі користувача є формування в пам'яті контейнера з даними типу Blob, з подальшим збереженням його на файлову систему користувача використовуючи механізм завантаження через елемент HTML «link».

В цілому можна сказати, що хоча на сьогодні відсутня єдина стандартизована технологія роботи з файлами в браузері, але завдяки гнучкості мови програмування JavaScript, можливо реалізувати необхідний інтерфейс, що буде працювати з деякими обмеженнями на розмір файлу в більшості браузерів.

3.3. Взаємодія з зовнішніми серверами та серверами ЦСК

При реалізації бібліотеки, що використовується в інфраструктурі відкритих ключів, виникає задача взаємодії з зовнішніми серверами ЦСК з використанням мови JavaScript. Складність вирішення цієї задачі полягає в необхідності виконання двох умов. По-перше, політикою безпеки браузера накладаються обмеження на запити до сторонніх серверів з використанням мови JavaScript. По-друге, сервери ЦСК мають жорсткі налаштування з безпеки та визначені стандартами формати передачі даних, зміна яких неможлива.

Існуючі технології WebSocket та CORS задовольняють першій вимозі, але їх використання неможливе у зв'язку з необхідністю модифікувати програмне забезпечення та налаштовувати сервери ЦСК. Забезпечити виконання обох вимог можливо лише за рахунок використання технології проксі-сервіса, який перенаправляє запити від клієнта до серверів ЦСК. Взаємодія з проксі-сервісом з JavaScript може відбуватися як з використанням XMLHttpRequest, так і технології WebSocket. Окрім можливості взаємодії з серверами ЦСК проксі-сервіс може виступати як брандмауер та фільтр прикладного рівня для надання доступу користувачів внутрішньої мережі до сервісів ЦСК без їх прямого підключення до мережі Інтернет.

3.4. Рекомендації з оптимізація роботи модуля

Робота інтерпретатора JavaScript в браузері організована таким чином, що за відображення сторінки та виконання коду мовою JavaScript відповідає один потік. Відомо, що криптографічні операції мають велику складність виконання, і їх використання безпосередньо на веб-сторінці призводить до зменшення відгуку сторінки на дії користувача. Для виконання обчислень, що потребують великого обсягу часу, в стандарті JavaScript було впроваджено інтерфейс WebWorkers, який дозволяє виконувати обчислення в іншому потоці та синхроні-

зувати дані з головним потоком програми. На сьогодні цю технологію вже підтримує більшість браузерів останніх версій, що дозволяє її використовувати.

При оптимізації швидкодії коду слід уникати використання та операцій з 64-бітними цілими числами, використання виключень C++. Браузери не підтримують 64-бітні цілі числа та емулюють операції з ними, а використання виключень C++ призводе до зниження ефективності оптимізації JavaScript коду браузером.

Компанією Intel, Mozilla та Google було анонсовано підтримку основних SIMD команд та типів в браузері з використанням бібліотеки SIMD.js. Поки це ще проект стандарту, та підтримується не всіма браузерами, але в компілятор Emscripten вже було додано можливість компілювати код, що використовує SIMD команди. Як було показано в роботі [8], за рахунок використання SIMD команд можна підвищити швидкодію в декілька разів.

Висновки

Проведений аналіз послуг безпеки інформації, які необхідно надавати в веб-додатках, дозволив визначити перелік вимог до криптографічної бібліотеки, що розробляється для використання в середовищі веб-браузера. На основі висунутих вимог було складено список критеріїв та проведено за ними порівняння існуючих технологій розробки.

В результаті порівняння технологій реалізації криптографічних бібліотек за критерієм кроссплатформеності бібліотека, реалізована за допомогою мови програмування JavaScript, зайняла перше місце, завдяки її роботі в браузерах лише з підтримкою інтерпретатора JavaScript та відсутності альтернативи в браузерах для мобільних платформ. За критеріями швидкодії, захисту від модифікації та управління безпекою бібліотеки, можливості роботи з апаратними носіями ключів та криптографічними модулями, а також універсальності рішення за сукупністю критеріїв перше місце зайняла реалізація бібліотеки з використанням технології Java Applet, що надає доступ до нативних бібліотек через інтерфейс взаємодії JNI. Хоча технологію Java Applet зараз підтримує більшість веб-браузерів для настільних комп'ютерів, відмова від її використання в веб-оглядачі Google Chrome в вересні 2015 року, наміри інших компаній припинити її використання та відсутність її підтримки в мобільних браузерах призводять до необхідності пошуку альтернативних рішень, таких, наприклад, як використання бібліотек JavaScript.

В рамках дослідження перспективного методу реалізації криптографічної бібліотеки за рахунок компіляції існуючих бібліотек мовою C/C++ в код мовою JavaScript було отримано практичні результати з швидкодії, які гірші за еталонну реалізацію мовою програмування C/C++ в 2 - 5 разів. Отримані результати підтверджують доцільність та перспективність подальшої роботи в цьому напрямку, а саме - дослідження підвищення швидкодії за рахунок використання паралельних обчислень. Також було запропоновано та реалізовано на практиці механізм доступу до серверів ЦСК безпосередньо з веб-браузера в умовах обмеження запитів до сторонніх серверів.

Список літератури: 1. *Oracle* Java Applets [Електронний ресурс]. - Режим доступу : http://www.oracle.com/technetwork/java/applets-137637.html 2. *Microsoft* Active X [Електронний ресурс]. - Режим доступу : https://msdn.microsoft.com/en-us/library/aa751972%28v=vs.85%29.aspx 3. *Mozilla* NAPI Plugin [Електронний ресурс]. - Режим доступу : https://developer.mozilla.org/en-US/Add-ons/Plugins/Gecko_Plugin_API_Reference 4. *Google* PPAPI [Електронний ресурс]. - Режим доступу : https://code.google.com/p/ppapi/wiki/GettingStarted 5. *ECMAScript* standarts [Електронний ресурс]. - Режим доступу : http://www.ecmascript.org/ 6. *Asm.js* [Електронний ресурс]. - Режим доступу : http://asmjs.org/ 7. *Emscripten* compiler [Електронний ресурс]. - Режим доступу : http://kripken.github.io/emscripten-site/ 8. *WebCrypto* API [Електронний ресурс]. - Режим доступу : http://www.w3.org/TR/WebCryptoAPI/ 7 File API [Електронний ресурс]. - Режим доступу : http://www.w3.org/TR/FileAPI/ 9. *Вплив* властивостей сучасних процесорів на продуктивність програм / О.Г. Качко, Д.С. Балагура, К.А. Погребняк // Прикладна радіоелектроніка. – 2014. – Т. 13, №3. – С. 181-285.

Харківський національний
університет радіоелектроніки

Надійшла до редколегії 15.03.2015