УДК 681.3:002.651.028(083.73)

Мелащенко А. О., Перевозчикова О. Л.

ТЕСТОВИЙ СТЕНД ДЛЯ ІНТЕРОПЕРАБЕЛЬНОСТІ ЕЛЕКТРОННИХ ЦИФРОВИХ ПІДПИСІВ

У статті ідентифіковано ключові блоки тестів ПТК у складі (А)ЦСК для побудови тестового стенда інтероперабельної НСЕЦП. На прикладі блоку верифікації відповідності ДСТУ ETSI TS 101 862 продемонстровано основні принципи побудови тестового стенда та їхню програмна реалізація.

Ключові слова: побудова тестового стенда, інтероперабельність електронних цифрових підписів.

Вступ

Одна з основних причин нинішньої відсутності інтероперабельності Національної системи електронних цифрових підписів (НСЕЦП) – прогалини нормативної бази, яка регулює технологічну та організаційну складові НСЕЦП. Наслідком цього є унеможливлення створення тестових стендів, що гарантують використання тільки інтероперабельних реалізацій програмно-технічних комплексів (ПТК) у НСЕЦП. Наявність простих помилок у реалізації форматів базових об'єктів, комплектів підписів, їхніх параметрів та інших компонентів є причиною неінтероперабельності НСЕЦП. Формальна методика акредитації ЦСК разом із тестовим стендом суттєво сприяли б вирішенню проблем інтероперабельності НСЕЦП. Детально проблеми НСЕЦП розглянуто в [1-3].

Закон України [4] реалізує Директиву 1999/93/ЄС [5], технологічно та організаційно підтриману стандартами, складовими еталонної моделі кваліфікованих цифрових підписів (QPKI). Нині діють гармонізовані ДСТУ, що підтримують Директиву 1999/93/ЄС і надають необхідну нормативну базу для створення формальної методики акредитації центрів сертифікації ключів (ЦСК). Також ці стандарти регламентують побудову специфікацій тестових стендів та їхню реалізацію для підвищення інтероперабельності ПТК у складі (А)ЦСК (тобто акредитованих ЦСК) [1].

Далі специфіковано вимоги до тестового стенду для підвищення інтероперабельності ПТК у складі (А)ЦСК у рамках НСЕЦП.

1. Загальна схема стенда

У цьому підрозділі деталізовано (розширено) вимоги до частини програмних тестів у складі стенду, описаного в [3]. На мал. 1 жирними лініями виділено блоки тестів, необхідних для гарантії інтероперабельної реалізації ПТК СА.

Для оцінки відповідності ПТК СА еталонній моделі QPKI [1] доцільні такі блоки тестів.

1. Профілі сертифікатів, списків скасованих сертифікатів (CRL) та інші протоколи. Формати цих об'єктів мають відповідати визнаним міжнародним стандартам ISO/IEC 9594-8:2008 [6], PKIX-профілів сертифікатів відкритих ключів та CRL (RFC 5280). Тести верифікують семантику та відповідні анотації про використання чи значення обмежень, тобто повну сумісність із ДСТУ ETSI TS 101 862:2009.

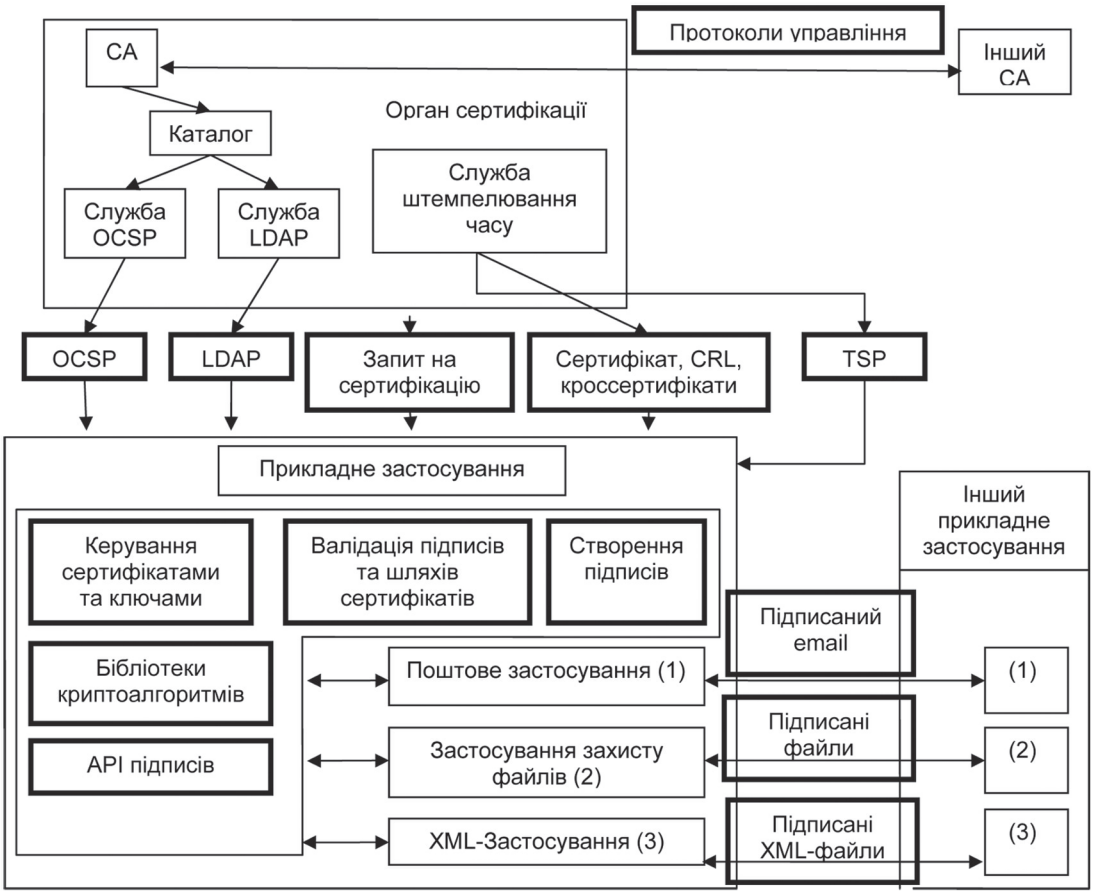
2. Управління PKI. Цей блок тестів верифікує онлайн-взаємодію між компонентами PKI, визначаючи профіль компонентів PKI, базованих на «Повідомленнях управління сертифікатами зі CMS» (RFC 5272, RFC 5273, RFC 5274).

3. Формати повідомлень, засновані на CMS [7]. Цей блок тестів перевіряє формати повідомлень, використовуваних в обміні даними між компонентами PKI. Формати мають відповідати ДСТУ ETSI TS 101 733:2009 та ДСТУ ETSI TS 101 734:2009.

4. Формати повідомлень, базовані на XML [8]. Цей блок тестів перевіряє формати повідомлень, використовувані в обміні даними між компонентами PKI відповідно до ДСТУ ETSI TS 101 903:2009 та ДСТУ ETSI TS 101 904:2009.

5. Функціональні протоколи, потрібні PKI для передачі сертифікатів, CRL або інформації про статус сертифіката за допомогою різноманітних застосувань (поштових клієнтів або Інтернет-браузерів). Завдання цього блоку тестів – вибрати «мінімальні достатні» за функціональністю репозитарії та методи доступу, здатні підтримати всі інтероперабельні репозитарії та клієнтські застосування, досягаючи взаємодії з автоматичною верифікацією підпису та шляхів

[1] Для позначення АЦСК і ЦСК використано також абревіатуру CA (Certification authority) як органа сертифікації.



Мал. 1. Блоки тестів у схемі взаємодії органів сертифікації та клієнтських застосувань

сертифікатів, незалежно від реалізації клієнта та сховища ПТК. Цей блок тестів базовано на найпоширеніших стандартах репозитаріїв, каталогах X.500 [9] та методах доступу, зазначених у PKIX Інтернет-стандартах: LDAP v3 та OCSP V1. Транспортні протоколи між репозитарієм і клієнтами обмежено TCP/IP (для LDAP) і HTTP (для OCSP).

6. Процедури валідації підписів і шляхів сертифікації. Цей блок тестів верифікує відповідність процедур валідації підписів і шляхів сертифікації згідно з проектом ДСТУ CWA 14171, а також підтримку політик підписування згідно з ETSI TR 102 272.

7. Криптографічні алгоритми. Цей блок тестів працює з переліком затверджених і рекомендованих криптоалгоритмів цифрових підписів, переважно рекомендацій ДСТУ ETSI TS 102 176:2009. Впродовж тестування перевіряють кросмодульну (між шаблонним криптомодулем та реалізованим) генерацію підпису (один криптоалгоритм генерує ЕЦП, другий – верифікує ЕЦП і навпаки).

8. API підпису. Цей блок тестів верифікує відповідність реалізованих API проектів ДСТУ EN 14890.

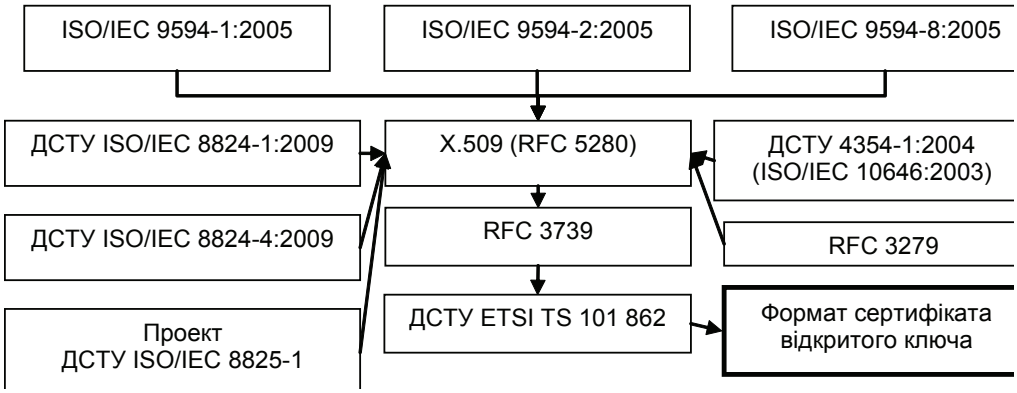
2. Приклад реалізації тестів верифікації реалізованого формату посиленого сертифіката

2.1. Розбір нормативних вимог

Перший блок тестів верифікує, зокрема, відповідність формату сертифікатів стандарту ETSI TS 101 862:2009, який спирається на ряд стандартів (див. мал. 2).

Користувачі відкритого ключа повинні бути впевнені в тому, що асоційований особистий ключ належить відправнику електронного документа (особі чи системі). Впевненість досягається за рахунок використання сертифікатів відкритих ключів (далі сертифікати) як структур даних, що зв'язують відкриті ключі зі суб'єктами. Зв'язок завіряє CA-видавець, підписуючи кожен сертифікат, що має обмежений термін дії, зазначений у підписаному контенті. Можливість незалежної перевірки клієнтом ЕЦП із сертифікату та періоду його валідності забезпечує поширення сертифіката по ненадійних каналах зв'язку і серверах, а також їхнього зберігання в ненадійному кеші систем, які використовують сертифікати.

ITU-T X.509 (раніше CCITT X.509) або ISO/IEC 9594-8, вперше опублікований у 1988 році



Мал. 2. Залежності між стандартами, що визначають вимоги до сертифікатів відкритих ключів

як частина рекомендацій каталогу X.500, визначає стандартний формат сертифіката X.509. Формат сертифіката в стандарті 1988 р. називають форматом версії 1 (v1). Після перегляду X.500 у 1993 р. додано два поля як результат зміни версії v1 на версію 2 (v2).

Специфікації Інтернет-пошти з посиленою конфіденційністю (PEM – Privacy Enhanced Mail) та опубліковані у 1993 р. RFC містять специфікації для PKI на основі X.509 v1 сертифікатів (RFC1422). З досвіду, накопиченого протягом розгортання RFC 1422, витікає неприйнятність форматів сертифікатів v1 і v2. Основний недолік – потреба в кількох полях для додаткових даних, цю потребу довела реалізація та досвід розгортання PEM. У відповідь на ці вимоги в ISO/IEC, ITU-T та ANSI X9 розробили формат сертифіката X.509 версії 3 (v3), що розширив формат v2, додавши опис додаткових полів розширень. Особливі види полів розширення можна вказати в стандартах або визначити і зареєструвати організаціям або товариствам. Стандартизацію основного формату версії 3 завершено в червні 1996 р.

Також ISO/IEC, ITU-T і ANSI X9 розробили стандартні розширення для використання в полях розширення версії 3. Вони можуть містити такі дані, як додаткові відомості про ідентифікацію суб'єкта, про атрибути ключа, про політику та обмеження шляху сертифікатів. Згідно з [4], в Україні допускаються лише чотирирівневий шлях. Наприклад, для клієнта українського банку шлях сертифікації виглядає як конкатенація даних із сертифікатів у такому ланцюжку суб'єктів НСЕЦП:

Клієнт банку – СА банку – Засвідчувальний центр Нацбанку України – ЦЗО.

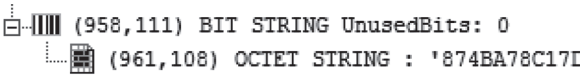
Зазначимо, що стандартні розширення ISO/IEC, ITU-T і ANSI X9 мають широке застосування. Для реалізації інтероперабельних X.509 v3 систем визначено профілі для використання X.509 v3 розширень. Наприклад, такий профіль визначено в RFC 5280.

Безпосередньо профіль X.509 сертифіката (RFC 5280) використовує фундаментальні стандарти інтероперабельної взаємодії (ДСТУ ISO/IEC 8824-1:2009, ДСТУ ISO/IEC 8824-4:2009, проект ДСТУ ISO/IEC 8825-1) і кодування інформації (ДСТУ 4354-1:2004). Оскільки профіль нейтральний до комплектів підписів, використовуваних для підписування сертифікатів, рекомендації з використання міжнародних комплектів підписів винесено в окремий документ (RFC 3279).

Загалом вимоги RFC 5280 профілює RFC 3739 (формально RFC 3280, але RFC 5280 замінив RFC 3280) на підтримку посилених сертифікатів. Вимоги RFC 3739 профілює ДСТУ ETSI TS 101 862 на підтримку посилених сертифікатів за визначеннями Директиви 1999/93/ЄС.

2.2. Підходи до реалізації специфікацій тестів

Побудова тестового стенду спирається на принципи, обов'язкові відповідно до реалізації Загальних критеріїв IT-захисту [10], що пов'язано з необхідністю недопущення неякісних реалізацій і заснованих на них претензіях. Приклад такої недобросовісної реалізації – просте вилучення всіх необхідних полів із сертифіката за допомогою стандартної/незалежної бібліотеки та наявності всіх полів, потрібних за нормативною базою. Цей підхід не проводить детального тестування. Наприклад, у кореневому сертифікаті ЦЗО поле підпису, яке має ASN.1 тип BIT STRING згідно з RFC 5280, містить фактичний підпис, закодований як тип OCTET STRING (мал. 3). Верифікація підпису не буде успішною, зважаючи на два додаткових значення, які прикріплено до значення підпису та ідентифікують тип OCTET STRING і його довжину.



Мал. 3. Закодований підпис ЦЗО

Для побудови тестових стендів доцільно використати такі три принципи.

1. Верифікація структури сертифіката. Стенд має верифікувати відповідність структури сертифіката ASN.1-нотаціям нормативної бази.

2. Верифікація комплектів підписів. Стенд має верифікувати підпис на сертифікаті, підпис із використанням сертифіката та кодування даних відповідно до контенту сертифіката та положень нормативної бази. Контент сертифіката містить параметри комплектів підписів для верифікації підпису на сертифікаті (рекомендовано використовувати самопідписані сертифікати та шлях сертифікатів). Під підписуванням із використанням сертифіката розуміють продукування та верифікацію підпису з використанням параметрів комплекту підписів та відкритого ключа, зазначеного в сертифікаті.

3. Верифікація контенту даних. Стенд має верифікувати контент полів і допустимі значення. Протягом верифікації контенту полів пропонується надавати стандартний набір реквізитів CA-видавця та суб'єкта для верифікації коректного заповнення відомостей про видавця й суб'єкта. Для верифікації допустимих значень необхідно верифікувати OID допустимих криптоалгоритмів, допустимі значення їхніх параметрів, наявність і контент розширень.

2.3. Приклад реалізації стенда

Прототип стенда реалізовано мовою Java [11], а бібліотеки для роботи з DER-закодованими ASN.1-нотаціями Bouncy Castle [12], тестові сертифікати згенеровано в операційному середовищі FreeBSD 8.0 [13] за допомогою утиліти OpenSSL [14].

Реалізацію принципу-1 розглянемо на прикладі верифікації нотації частини сертифіката RFC 5280 (див. лістинг 1).

```
Certificate ::= SEQUENCE {
    tbsCertificate       TBSCertificate, -- Контент
  сертифікату
 -- Комплект підпису та його параметри
       signatureAlgorithm   AlgorithmIdentifier,
   signature   BIT STRING -- Значення підпису}
```

Лістинг 1. Частина структури сертифіката RFC 5280

Розглянемо приклад верифікації сертифікату (лістинг 1), відображений у лістингу 2.

1	public static void main(String[] args) throws FileNotFoundException, IOException {
2	FileInputStream fInput = new FileInputStream(args[0]);
3	ASN1InputStream input = new ASN1InputStream(fInput);
4	DERObject obj = input.readObject(); // отримуємо перший об'єкт сертифікату
5	DERSequence certificate = null; // створюється змінна для вміщення першого об'єкта з потоку // батів
6	if (obj instanceof DERSequence) {
7	certificate = (DERSequence) obj;
8	} else {
9	System.out.println(«Ошибка 1, элемент на является ASN.1 типом SEQUENCE»); System.exit(1); }
10	DERSequence tbsCertificate = null;
11	DERSequence signatureAlgorithm = null;
12	DERBitString signature = null;
13	if (certificate.getObjectAt(0) instanceof DERSequence) {
14	tbsCertificate = (DERSequence) certificate. getObjectAt(0);
15	} else {
16	System.out.println(«Ошибка 1, элемент на является ASN.1 типом SEQUENCE»); System.exit(1); }
17	if (certificate.getObjectAt(1) instanceof DERSequence) {
18	signatureAlgorithm = (DERSequence) certificate. getObjectAt(1);
19	} else {
20	System.out.println(«Ошибка 1, элемент на является ASN.1 типом SEQUENCE»); System.exit(1); }
21	if (certificate.getObjectAt(2) instanceof DERBitString) {
22	signature = (DERBitString) certificate.getObjectAt(2);
23	} else {
24	System.out.println(«Ошибка 2, элемент на является ASN.1 типом BITSTRING»); System.exit(2);
25	System.out.println(«Все тесты пройдены успешно»);

Лістинг 2. Реалізація верифікації нотації сертифіката

1) У 1-у рядку визначено ім'я функції та її параметри.

2) У рядках 2 і 3 створено вхідний потік байтів, що подає DER-закодований сертифікат ЦЗО. Далі під усіма типами маємо на увазі DER-закодовані типи ASN.1-нотацій.

3) Нотація Certificate, згідно з RFC 5280 (лістинг 1), має тип SEQUENCE, тобто перший об'єкт потоку байтів має тип SEQUENCE. У рядку 6 це підтверджено, у разі успішного виконання умови змінній certificate (рядок 7) присвоєно перший об'єкт потоку байтів, інакше виводиться повідомлення про помилку і виходимо з програми.

4) Згідно з RFC 5280 (лістинг 1), нотації TBSCertificate та AlgorithmIdentifier мають тип SEQUENCE, для них виконують дії, аналогічні кроку 3. Створення змінних (рядки 10 і 11), перевірка типів (рядки 13 і 17), у разі успішної перевірки змінним присвоєно об'єкти (рядки 14 і 18 відповідно), інакше виводимо помилки (рядки 16 і 20 відповідно).

5) Перевірку типу нотації signature проводимо аналогічно кроку 3. Створено змінну (рядок 12), перевірено тип (рядок 21), у разі успішної перевірки змінним присвоєно об'єкти (рядок 22), інакше виводимо повідомлення про помилку (рядок 24). Але підпис сертифіката ЦЗО міститься в типі OCTET STRING (нагадаємо примітку), поміщеному в тип BIT STRING, що не припустимо (мал. 3). Зазначимо, що пройдено успішно всі тести часткової верифікації нотацій сертифіката ЦЗО.

6) У рядку 25 виводимо повідомлення про успішне проходження тесту.

У прикладах реалізації принципів 2 і 3 використано сертифікати, згенеровані за допомогою утиліти OpenSSL. Це обумовлено інтегрованою підтримкою міжнародних комплектів підписів мовою Java та недоцільністю перевантаження коду лістингів складними перетвореннями. Для підписування сертифікатів CA у першій черзі свого ПТК, що діє до кінця 2010 р., ЦЗО підтримує тільки два криптоалгоритми: застарілий ГОСТ 34.310-95 та ДСТУ 4145:2002, що значно ускладнило впровадження електронного документообігу в Україні, заснованого на ЕЦП. У разі використання нових сертифікатів буде продемонстровано наслідки перетворення підпису із сертифіката на тип OCTET STRING.

Згенеровані сертифікати є самопідписаними сертифікатами за допомогою sha1-with-RSA, OID 1.2.840.113549.1.1.5. Згідно з RFC 3279, параметри комплекту підпису нотації AlgorithmIdentifier мають бути відсутніми, а поле відкритого ключа нотації SubjectPublicKeyInfo відповідати нотації:

```
RSAPublicKey ::= SEQUENCE {
    modulus           INTEGER,    -- n
    publicExponent    INTEGER  }  -- e
```

Приклад тестів реалізації принципу-2 відображено в лістингу 3.

26	X509CertificateStructure cert = new X509Certificate Structure(certificate);
27	
28	SubjectPublicKeyInfo pubInfo = cert. getTBSCertificate().getSubjectPublicKeyInfo();
29	if(!pubInfo.getAlgorithmId().getObjectId().getId(). equals(«1.2.840.113549.1.1.5»)){
30	System.out.println(«Невалидное значение алгоритма подписания»); System.exit(3);}
31	if(!(pubInfo.getAlgorithmId().getParameters() instanceof DERNull)){
32	System.out.println(«Параметры алгоритма должны отсутствовать»); System.exit(3);}
33	DERSequence pubSeq = (DERSequence) pubInfo. getPublicKey();
34	if(!(pubSeq.getObjectAt(0) instanceof DERInteger)){
35	System.out.println(«Первая состовляющая открытого ключа должна иметь тип INTEGER»); System.exit(3);}
36	if(!(pubSeq.getObjectAt(1) instanceof DERInteger)){
37	System.out.println(«Вторая состовляющая открытого ключа должна иметь тип INTEGER»); System.exit(3);}
38	
39	KeyFactory keyFactory = KeyFactory. getInstance(«RSA»);
40	File privFile = new File(«key.der»);
41	DataInputStream dis = new DataInputStream(new FileInputStream(privFile));
42	byte[] privKeyBytes = new byte[(int) privFile. length()];
43	dis.read(privKeyBytes);
44	dis.close();
45	
46	X509EncodedKeySpec pubSpec = new X509EncodedKeySpec(cert. getSubjectPublicKeyInfo().getDEREncoded());
47	RSAPublicKey pubKey = (RSAPublicKey) keyFactory.generatePublic(pubSpec);
48	PKCS8EncodedKeySpec privSpec = new PKCS8Enc odedKeySpec(privKeyBytes)
49	RSAPrivateKey privKey = (RSAPrivateKey) keyFactory.generatePrivate(privSpec);
50	
51	Signature sig = Signature. getInstance(«SHA1withRSA»); // создает объекта для подписания и //верификации подписи, используя параметры сертификата
52	sig.initSign(privKey); // загружает личный ключ
53	byte[] message = new byte[]{(byte) 't', (byte) 'e', (byte) 's', (byte) 't'}; // создает тестовое // сообщения для подписания
54	sig.update(message); // сообщение помещается в объект для последующего хеширования
55	byte[] sigBytes = sig.sign(); // генерируется подпись
56	sig.initVerify(pubKey);
57	sig.update(message);
58	if (sig.verify(sigBytes)) {
59	System.out.println(«Подпись верифицированна»);
60	} else {
61	System.out.println(«Подпись НЕ верифицированна»); System.exit(3);}
62	
63	sig.update(cert.getTBSCertificate(). getDEREncoded());
64	if (sig.verify(cert.getSignature().getBytes())) {
65	System.out.println(«Подпись на сертификате верифицированна»);
66	} else {
67	System.out.println(«Подпись на сертификате НЕ верифицированна»); System.exit(3);}

Лістинг 3. Верифікація криптографічного перетворення коду ЕЦП

1) Для зручності маніпулювання даними після блоку тестів, який верифікував відповідність сертифіката нотації, використаємо стандартні

структури. Згідно з рядком 26, поміщаємо DER-закодований сертифікат у відповідну об'єктну модель сертифіката.

2) Перед використанням параметрів комплекту підпису для підписування і верифікації об'єктів перевіряємо їх. Для зручності в 28 рядку поле сертифіката, що відповідає за параметри комплекту підпису, поміщаємо в змінну pubInfo.

3) Рядки 29 і 30 порівнюють OID алгоритму підписування з шаблонним, при відмінних від шаблонних значень видаємо помилку.

4) У рядках 31 і 32 перевіряємо відсутність параметрів у полі parameters нотації AlgorithmIdentifier, за її наявності видаємо повідомлення про помилку.

5) Рядки з 33 по 37 верифікують виконання вимог нотації SubjectPublicKeyInfo.

6) Після успішної верифікації параметрів комплекту підпису перевіряємо відповідність відкритого ключа особистому (цей крок опущено). Рядки 39-49 завантажують особистий і відкритий ключі у відповідні структури для подальшого використання.

7) Рядки 57-61 завантажують відкритий ключ, верифікують підпис і виводять повідомлення про результат верифікації; при успішнфій верифікації тести гарантують коректність параметрів комплекту підпису для використання в ЕЦП.

8) У наступному кроці верифікуємо ЕЦП самого сертифіката, оскільки використано самопідписаний сертифікат. Відкритий ключ для верифікації ЕЦП сертифіката міститься в полі subjectPublicKey нотації SubjectPublicKeyInfo. Рядок 63 поміщає нотацію TBSCertificate в об'єкт верифікації для обчислення геша. Саме значення нотації TBSCertificate підписано в сертифікаті.

9) У рядках 64-67 відображено верифікацію ЕЦП сертифіката, що аналогічно кроку 7.

У RFC 3739 профільовано вимоги RFC 5280, зокрема встановлено вимоги до формату полів нотації Issuer: domainComponent; countryName; stateOrProvinceName; organizationName; localityName; serialNumber. ДСТУ ETSI TS 101 862 профілює RFC 3739 і встановлює вимоги до обов'язкових полів розширень сертифіката. Приклад тесту на наявність розширення сертифіката розглянемо з використанням розширення basicConstraint сертифіката (див. лістинг 4).

68	X509Extension basicConstraint = cert.getTBSCertificate().getExtensions().getExtension(new DERObjectIdentifier(«2.5.29.19»));
69	DEROctetString value = (DEROctetString) basicConstraint.getValue();
70	DERSequence valueSeq = (DERSequence) DERSequence.fromByteArray(value.getOctets());
71	DERBoolean v = (DERBoolean) valueSeq.getObjectAt(0);
72	if(!v.isTrue()){
73	System.out.println(«Невалидное значение расширения basicConstraint»); System.exit(4);}
74	
75	public static boolean checkExtention(X509Certificate Structure cert, String oid, String defaultValue){
76	Vector val = cert.getIssuer().getValues(new DERObjectIdentifier(oid));
77	if(val.size() > 0){
78	String str = (String) val.get(0);
79	str = new String(str.getBytes(Charset.forName(«UTF8»)));
80	if(!str.equals(value)){
81	System.out.println(«Невалидное значение или его кодировка»); return false;} return true;}
82	return false;}
83	
84	if(!checkExtention(cert, «2.5.4.6», «UA»)){System.exit(5);} //countryName
85	if(!checkExtention(cert, «2.5.4.8», «Kiev»)){System.exit(5);} //stateOrProvinceName
86	if(!checkExtention(cert, «2.5.4.10», «Test Org»)){System.exit(5);} //organizationName
87	if(!checkExtention(cert, «2.5.4.7», «Kiev»)){System.exit(5);} //localityName
88	if(!checkExtention(cert, «0.9.2342.19200300.100.1.25», «test.org»)){System.exit(5);} //domainComponent
89	input.close();
90	fInput.close();}

Лістинг 4. Реалізація тестів структуру поля Issuer і наявності розширення сертифікати

1) Згідно з рядком 68, для зручності маніпулювання значення розширення сертифіката поміщаємо в змінну basicConstraint.

2) Згідно з RFC 5280, розширення сертифіката є парами типу SQUENCE, складеними з двох значень типів OID і OCTET STRING. У рядках 69-71 декодовано значення типу BOOLLEAN з OCTET STRING.

3) У рядках 72-73 перевіряємо значення розширення, якщо воно не TRUE, виводимо помилку.

4) Наступний блок тестів верифікує наявність всіх складових, що ідентифікують CA-видавця сертифіката. Оскільки перевірка ідентична, у рядках 75-82 визначено функцію, яка верифікує наявність і контент складових.

5) Рядок 76 отримує значення складової.

6) Наявність значення складової верифікуємо в рядку 77.

7) Рядки 78-81 верифікують коректність кодування значень, у разі використання кодування, відмінного від UTF-8, видаємо помилку, і функція повертає FALSE. Зазначимо використання «шаблонів підходу», тобто для створення сертифіката розробник використовує стандартні значення полів, що дозволяє стенду верифікувати коректність їхнього заповнення.

8) Рядки 84-88 використовують раніше визначену функцію для верифікації наявності та контенту складових нотації Issuer. Параметрами функції є OID складової та шаблонне значення. У разі невдалої перевірки виходимо з програми з відповідним кодом помилки.

9) Рядки 89-90 закривають потік байтів і файл відповідно.

Висновки

Фактично не вдалися спроби домогтися інтероперабельності реалізацій ПТК у складі (А) ЦСК організаційними заходами, тобто затвердженням набору регуляторних актів Держкомінформатизації України. Неможливо довести неінтероперабельність реалізації ПТК без його фактичної перевірки на програмно реалізованому тестовому стенді.

Доцільно використовувати специфікації програмних тестів на основі нормативних документів, насамперед гармонізованих ДСТУ, і подальші реалізації, підтримані організаційними заходами вирішення спорів. Це сприяє не тільки досягненню інтероперабельності НСЕЦП, а й подальшій кроссертифікації.

1. Мелащенко А. О. Проблемы интероперабельности Национальной системы электронных цифровых подписей / Мелащенко А. О., Перевозчикова О. Л. // Кибернетика и системный анализ. – 2009. – № 6. – С. 55–63.
2. Мелащенко А. О. Комплекти підписів для інтероперабельності Національної системи електронних цифрових підписів / А. О. Мелащенко, О. Л. Перевозчикова, О. С. Скарлат, К. С. Криворучко // Наукові записки. НаУКМА. Т 99 : Комп'ютерні науки. – К., 2010. – С. 70–77.
3. Мелащенко А. О. Кроссертификация Украины / Мелащенко А. О., Перевозчикова О. Л. // Проблемы программирования. – 2010. – №2–3. – С. 299–308.
4. Закон України «Про електронний цифровий підпис» від 22.05.2003 № 852 [Електронний ресурс]. – Режим доступу : URL : http://zakon.rada.gov.ua/cgi-bin/laws/main.cgi?nreg=852-15. – Назва з екрана.
5. Директива 1999/93/ЄС про комунікаційне середовище електронних підписів від 13 грудня 1999 р.
6. ISO/IEC 9594-8:2008 Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services [Електронний ресурс]. – Режим доступу : URL : http://www.iso.org/iso/catalogue_detail.htm?csnumber=41884. – Назва з екрана.
7. W3C Recommendation 26 November 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Електронний ресурс]. – Режим доступу : URL : http://www.w3.org/TR/2008/REC-xml-20081126/. – Назва з екрана.
8. RFC 3852 Cryptographic Message Syntax (CMS) [Електронний ресурс]. – Режим доступу : URL : http://tools.ietf.org/html/rfc3852. – Назва з екрана.
9. ISO/IEC 9594-1:2008 Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks [Електронний ресурс]. – Режим доступу : URL : http://www.iso.org/iso/catalogue_detail.htm? csnumber=53372. – Назва з екрана.
10. ISO/IEC 15408:2009 Information technology – Security techniques – Evaluation criteria for IT security [Електронний ресурс]. – Режим доступу : URL : http://www.iso.org/iso/catalogue_detail.htm?csnumber=40612. – Назва з екрана.
11. java.sun.com
12. http://www.bouncycastle.org
13. www.freebsd.org
14. www.openssl.org

A. Melashchenko, O. Perevozchikova

TESTBED FOR INTEROPERABILITY OF DIGITAL SIGNATURES

The paper identified key building blocks of tests of CA Trustworthy system for construction testbed of interoperable NSES (National system for electronic signature). Verification block of compliance DSTU ETSI TS 101 862 demonstrated the basic principles of the test stand and its software implementation.

Keywords: electronic signature, interoperability, test stand, certificate of the public key, verification.