



УКРАЇНА

(19) UA (11) 84861 (13) C2  
(51) МПК (2006)  
G07B 17/00

МІНІСТЕРСТВО ОСВІТИ  
І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ДЕПАРТАМЕНТ  
ІНТЕЛЕКТУАЛЬНОЇ  
ВЛАСНОСТІ

## ОПИС ДО ПАТЕНТУ НА ВІНАХІД

### (54) СПОСІБ ПЕРЕВІРКИ СПРАВЖНОСТІ ПОЗНАЧКИ ПРО СПЛАТУ ПОШТОВОГО ЗБОРУ

1

(21) а200507603  
(22) 21.01.2004  
(24) 10.12.2008  
(86) РСТ/DE2004/000083, 21.01.2004  
(31) 103 05 730.7  
(32) 12.02.2003  
(33) DE  
(46) 10.12.2008, Бюл.№ 23, 2008 р.  
(72) ФЕРІ ПЕТЕР, ГЕЛЬМУС ЮРГЕН, МАЙЕР  
ГЮНТЕР, ШТУММ ДІТЕР, ФУЛЬРІДЕ КАРСТЕН  
(73) ДОЙЧЕ ПОСТ АГ  
(56) WO 01/17160 A1, 08.03.2001  
EP 0854444 A2, 22.07.1998  
US 5150408 A, 22.09.1992  
EP 0735722 A2, 02.10.1996  
(57) 1. Спосіб перевірки справжності позначки  
про сплату поштового збору, проставленої на  
поштовому відправленні за допомогою франку-  
вального ключа, який полягає в дешифруванні  
криптографічної інформації, що міститься у по-  
значці, і в користуванні нею для перевірки  
справжності цієї позначки, який **відрізняється**  
тим, що утворюють ключ даних і передають його  
від центральної системи забезпечення платежів  
до локальних систем забезпечення платежів, які  
його імпортують, а результат імпортування  
передають до центральної системи забезпечення  
платежів і після успішного імпортування ключа  
даних суттєво всіма локальними системами  
забезпечення платежів ключ даних (КД)  
звільняють для формування нової позначки.  
2. Спосіб за п.1, який **відрізняється** тим, що до  
ключа даних додають кінцеву дату для попере-  
днього ключа і/або його передають разом з цією  
кінцевою датою.  
3. Спосіб за п.2, який **відрізняється** тим, що  
франкувальний ключ передають до криптографі-  
чного елемента, який виконує перевірку, чи ма-  
ють інші ключі даних кінцеву дату і чи кінцева  
дата, збережена для наступного ключа даних,  
передусь дати, збереженій у системі забезпечення  
платежів.  
4. Спосіб за будь-яким з попередніх пунктів, який  
**відрізняється** тим, що до ключа даних додають  
лічильник версій і/або його передають разом з  
цим лічильником версій.  
5. Спосіб за п.4, який **відрізняється** тим, що  
ключ даних передають до криптографічного еле-

2

мента, зокрема криптокарти, який (яка) негайно  
після прийому ключа даних знищує всі ключі да-  
них, які мають таке саме значення лічильника  
версій, як переданий ключ даних.  
6. Спосіб за будь-яким з попередніх пунктів, який  
**відрізняється** тим, що результат імпортування  
передають як запис даних.  
7. Спосіб за п.6, який **відрізняється** тим, що за-  
пис даних містить ключ.  
8. Спосіб за будь-яким з попередніх пунктів, який  
**відрізняється** тим, що результат імпортування  
передають до центра перенесення вартості по-  
штового відправлення.  
9. Спосіб за будь-яким з попередніх пунктів, який  
**відрізняється** тим, що перевірку результату ім-  
портування здійснюють дешифруванням ключа.  
10. Спосіб за будь-яким з попередніх пунктів,  
який **відрізняється** тим, що після успішного ім-  
портування ключа даних у суттєво всі локальні  
системи забезпечення платежів цей ключ даних  
звільняють у центрі перенесення вартості пошто-  
вого відправлення.  
11. Спосіб за будь-яким з попередніх пунктів,  
який **відрізняється** тим, що ключ даних є симет-  
ричним ключем.  
12. Спосіб за будь-яким з попередніх пунктів,  
який **відрізняється** тим, що новий ключ даних  
передають від центра перенесення вартості по-  
штового відправлення до центральної системи  
забезпечення платежів.  
13. Спосіб за будь-яким з попередніх пунктів,  
який **відрізняється** тим, що у центрі перенесен-  
ня вартості поштового відправлення шифрують  
ключ даних транспортним ключем.  
14. Спосіб за п.13, який **відрізняється** тим, що  
транспортний ключ шифрують головним транс-  
портним ключем.  
15. Спосіб за будь-яким з попередніх пунктів,  
який **відрізняється** тим, що ключ даних утворю-  
ють у зоні центра перенесення вартості поштово-  
го відправлення.  
16. Спосіб за будь-яким з попередніх пунктів,  
який **відрізняється** тим, що до ключа даних до-  
дають інформацію для його розпізнавання.  
17. Спосіб за п.16, який **відрізняється** тим, що  
інформацію для розпізнавання ключа передають  
в шифрованій формі.

(13) C2

(11) 84861

(19) UA

18. Спосіб за будь-яким з попередніх пунктів, який **відрізняється** тим, що ключ даних або щонайменше частина ключа даних є компонентом

франкувального ключа для створення позначки про сплату поштового збору.

Винахід стосується способу верифікації аутентичності поштового штемпелю, генерованого з використанням франкувального ключа і застосованого до поштової кореспонденції, причому у процесі застосування цього способу криптографічна інформація, яка міститься у поштовому штемпелі, дешифрується і використовується для верифікації аутентичності цього поштового штемпелю.

Існують відомі процедури забезпечення поштових кореспонденцій цифровим поштовим штемпелем.

Для полегшення отримання поштового штемпелю відправником поштової кореспонденції можна, наприклад, застосувати франкувальну систему, що використовується у Deutsche Post AG, отримувати поштові штемпелі у системі клієнта і надсилати їх на принтер через будь-який інтерфейс.

Спосіб такого типу описано у [заявці DE 1000 20 402 A1].

Для виключення шахрайського використання цього методу, цифровий поштовий штемпель містить криптографічну інформацію, наприклад, про дійсність системи клієнта, яка контролює продукування поштових штемпелів.

[У міжнародній заявці WO 01/17160 A1] описано спосіб розподілення ключів, які генеруються центральним вузлом менеджменту і передаються у зашифрованій формі розподільчому вузлу до шифрувальних пристроїв. Ці пристрої через розподільчому вузол надсилають повідомлення про успішний або неуспішний прийом ключа до вузла менеджменту. Якщо прийом ключа є неможливим, інформація про це знову передається у нешифрованій формі до належного шифрувального пристрою.

[В Європейському патенті EP 0 854 444 A2] описано спосіб шифрування і перевірки поштового штемпелю, сформованого франкувальною машиною. Тут на основі головного ключа, застосованого у франкувальній машині, будуються додаткові ключі, які використовують для формування поштового штемпелю. У поштовому центрі подібним чином генеруються додаткові ключі, які використовуються для перевірки поштових штемпелів.

[У патенті США 5 150 408] описано спосіб розподілення ключів у системі зв'язку, наприклад, безпроводній мережі, коли шифроване повідомлення передається до пристрою зв'язку вузлом розподілення ключів. Після прийому цього повідомлення пристрій зв'язку надсилає підтвердження до вузла розподілення ключів.

Існує стандарт ANSI X9.17 передачі шифрованих даних [див. URL [http://csrc.nist.gov/publications/nidtpubs/800-](http://csrc.nist.gov/publications/nidtpubs/800-7/node209.htm)

[7/node209.htm](http://csrc.nist.gov/publications/nidtpubs/800-7/node209.htm) від 7/10/1994], який базується на ієрархії серед декількох ключів. Тут існує щонайменше один ключ даних короткої тривалості, обмін яким електронним шляхом у зашифрованій формі здійснюється між партнерами зв'язку, як це відбувається також і з ключем для шифрування ключа даних, обмін яким здійснюється вручну. У даному документі описано відомий спосіб Діффі-Гелмана (Diffie-Hellman) шифрування даних, згідно з яким партнери зв'язку обчислюють спільний ключ з відомих чисел і з секретного випадкового числа.

[В Європейському патенті EP 0 735 722 A2] описано систему розпорядження ключами для генерування, розподілення і контролю ключів для франкувальних машин. Тут існують декілька убезпечених зон, з'єднаних з комп'ютерами, які дозволяють здійснювати зв'язок між зонами і контроль цих зон. Кожна з операцій генерування ключів, їх встановлення, верифікації і підтвердження виконується в одній з цих зон. Кожна зона має зв'язок з архівом, в якому протоколюється статус зони.

Задачею винаходу є створення способу швидкої і надійної верифікації аутентичності поштового штемпелю. Зокрема, цей спосіб має бути придатний для широкого застосування у процедурах верифікації, у першу чергу у поштових або вантажних центрах.

Згідно з винаходом, цю задачу вирішено, як це визначено у п.1 Формули винаходу.

Зокрема спосіб верифікації аутентичності реалізується таким чином, що ключ даних генерується і передається центральною базою даних (центральна система ZinS) до локальних систем убезпечення платежів (локальна ZinS). Це здійснюється шляхом використання генерованого франкувального ключа і поштового штемпелю, застосованих до поштової кореспонденції, завдяки чому криптографічна інформація, що міститься у поштовому штемпелі, дешифрується і використовується для верифікації аутентичності цього поштового штемпелю.

Для підвищення безпечності способу доцільно, щоб локальна система убезпечення платежів імпортувала ключ даних і передавала результат імпортування до центральної бази даних (центральної системи ZinS).

У найбільш бажаному втіленні винаходу результат імпортування передається у вигляді запису даних.

Бажано, щоб запис даних містив ключ. Цей ключ може мати різні атрибути. Зокрема, він може бути симетричним або асиметричним ключем. Залежно від призначення ключ слугує для дешифрування і/або шифрування інформації. За своєю природою ключ може також переносити інфо-

рмацію. Наприклад, ключ може містити інформацію про франкувальний ключ і дату його генерування і/або закінчення дійсності.

Для забезпечення однорідного обміну ключами доцільно, щоб центральна база даних (центральна система ZinS) перевіряла результат імпортування і передавала його до центру передачі вартостей (Поштового Пункту). Це втілення винаходу уможливило виконання у центрі передачі вартостей такої подальшої операції процесу, як визначення результату імпортування.

Результат можна перевіряти різними шляхами. Однак, це найкраще робити дешифруванням ключа.

Бажане втілення винаходу характеризується тим, що після успішного імпортування ключа даних майже в усі локальні системи забезпечення платежів (локальні ZinS) ключ даних вивільняється як новий франкувальний ключ у центрі передачі вартостей (Поштовому Пункті) і у подальшому використовується для формування нового поштового штемпелю.

Це втілення способу дозволяє виконувати заміну ключів у центрі передачі вартостей як функцію попередніх заміни ключів у системах забезпечення платежів. Цим забезпечується те, що поштовий штемпель буде сформований лише з використанням нового ключа, якщо цей новий ключ вже є у локальних системах забезпечення платежів. Цим забезпечується можливість верифікації дійсності кожного сформованого поштового штемпелю локальною системою забезпечення платежів.

Бажане втілення способу згідно з винаходом з застосуванням контролю заміни ключів у центрі передачі вартостей як функції заміни ключів у локальній системі забезпечення платежів є особливо зручним у сполученні з щонайменше однією з інших операцій процесу.

Зокрема, сполучення декількох операцій забезпечує швидкий і надійний обмін ключами з верифікацією кожного обміну.

При заміні ключів доцільно передавати нові ключі даних від центра передачі вартостей (Поштового Пункту) до центральної системи забезпечення платежів.

Особливо зручно шифрувати ключ даних транспортним ключем (ТК) у центрі передачі вартостей.

Доцільно шифрувати транспортний ключ головним транспортним ключем (ГТМ).

Бажано генерувати ключ даних у зоні центра передачі вартостей. Цим виключаються шахрайські заміни ключів даних під час їх передачі до центра передачі вартостей.

Для додаткового підвищення захисту даних доцільно забезпечувати ключ даних ідентифікаційною інформацією про ключ.

Бажано передавати цю ідентифікаційну інформацію у зашифрованій формі.

Щоб забезпечити використання дійсного ключа даних у кожній з операцій шифрування і/або дешифрування, бажано додавати до ключа даних лічильник генерувань і передавати його з цим лічильником.

Щоб уникнути використання недійсних ключів, бажано додавати до франкувального ключа кінцеву дату попереднього ключа даних і/або передавати його разом з цією кінцевою датою.

Описаний ключ даних можна використовувати для однієї або більше часткових верифікацій, а також для формування поштового штемпелю і/або для дешифрування інформації, що міститься у цьому штемпелі.

Бажано, щоб часткова верифікація включала дешифрування криптографічної інформації, що міститься у поштовому штемпелі.

Інтегрування дешифрування криптографічної інформації у процес верифікації дозволяє безпосередньо підтвердити аутентичність поштового штемпелю, завдяки чому верифікацію можна проводити у реальному часі.

Крім того, бажано, щоб одна з часткових верифікацій включала порівняння між поточною датою і датою формування поштового штемпелю. Додання дати формування поштового штемпелю, особливо у шифрованій формі, підвищує захищеність даних, оскільки порівняння між поточною датою і датою формування поштового штемпелю відвертає багаторазове використання поштового штемпелю.

Для підвищення швидкості верифікації бажано, щоб вузол зчитування і вузол верифікації обмінювались інформацією з застосуванням синхронного протоколу.

В іншому бажаному втіленні винаходу вузол зчитування і вузол верифікації має зв'язок з застосуванням асинхронного протоколу.

Особливо зручно телеграфувати дані від вузла зчитування до вузла верифікації.

Інші переваги, особливі ознаки і практичні удосконалення визначено у залежних п.п.Формули і розглядаються у наведеному нижче описі з посиланнями на креслення, в яких:

Фіг.1 - бажане втілення розподілення ключів згідно з винаходом,

Фіг.2 - варіанти застосування розподілення ключів згідно з винаходом,

Фіг.3 - схема інтерфейсу між центральною системою забезпечення платежів (центральною ZinS) і центром передачі вартостей (Поштовим Пунктом),

Фіг.4 - операції процесу інтегрування ключа даних у центральну систему забезпечення платежів (центральну ZinS),

Фіг.5 - схема розподілення ключів від центральної системи забезпечення платежів (центральною ZinS) до локальних систем забезпечення платежів, включаючи компетентні криптосистеми,

Фіг.6 - зв'язок з інтерфейсом DDL,

Фіг.7 - схема операцій процесу інкапсуляції компонентів і обробки хибних повідомлень.

Далі наведено опис винаходу з посиланням на систему франкування РС. Операції забезпечення платежів є незалежними від системи формування поштових штемпелів.

Особливо бажаною є децентралізована верифікація на індивідуальних верифікаційних станціях, зокрема, у поштових центрах, але припустимою є також централізована верифікація.

У бажаному втіленні винаходу індивідуальними верифікаційними станціями є локальні системи забезпечення платежів, з'єднані, бажано, з центральною системою забезпечення платежів зв'язком для передачі даних.

В іншому втіленні центральною системою забезпечення платежів (центральною ZmS) є Поштовий Пункт.

Інтерфейс між центром передачі вартостей і системою забезпечення платежів складається з криптографічних ключів.

Ключ, яким мають обмінюватись центр передачі вартостей і система забезпечення платежів, забезпечує захист поштового штемпелю від підробки. Це досягається тим, що частина штрих-коду 2D, який утворює поштовий штемпель, зашифровується зазначеним ключем. Оскільки з технологічних міркувань вибраний ключ є симетричним, він має бути переданий від центра передачі вартостей до системи забезпечення платежів, де він розподіляється між індивідуальними поштовими центрами.

Надійне зберігання ключів забезпечується використанням спеціальних криптокарт. Винахід включає декілька застосувань, в яких ці ключі використовуються з застосуванням спеціальних технічних засобів. Повний життєвий цикл цих ключів - генерування, експорт, розподілення і нарешті імпорту у децентралізовану систему - є необхідним для оптимізації параметрів системи.

Фіг 1 ілюструє бажану послідовність для розподілення ключів, а саме, бажане розподілення між центром передачі вартостей і центральною системою забезпечення платежів.

Першою операцією 1 заміни франкувального ключа, призначеного для формування поштового штемпелю, є генерування нового франкувального ключа. Термін "франкувальний ключ" тут не слід розуміти в обмежувальному сенсі, оскільки він може бути використаний багатьма шляхами для формування поштового штемпелю.

Наприклад, франкувальний ключ можна використовувати для формування поштового штемпелю безпосередньо.

Є можливим і бажаним, однак, у системах підвищеної захищеності проти маніпуляцій формувати поштовий штемпель з мультишифрованим вмістом даних, причому цей вміст бажано формувати як результат декількох операцій в одному або декількох місцях і вводити результат операції у поштовий штемпель франкувальним ключем.

Наприклад, у франкувальний ключ вводять крипторядок типу, описаного у [DE 100 20 402 A1].

Для подальшого захисту проти шахрайського виготовлення поштових штемпелів застосовується заміна франкувального ключем, специфічна для певного місця.

У даному випадку франкувальний ключ ново генерують з регулярним інтервалом, наприклад, після закінчення заздалегідь визначеного інтервалу часу.

Можна також виконувати нове генерування франкувального ключа як функцію інших параме-

трів, наприклад, на основі завантаженої суми поштових зборів і/або франкованих поштових кореспонденцій.

Взагалі нове генерування нового франкувального ключа можна виконувати у будь-якому місці, однак, для кращої безпеки бажано мінімізувати дії, пов'язані з передачею нового франкувального ключа і генерувати його у центрі передачі вартостей або у зоні цього центра.

Для забезпечення високого рівня захисту проти шахрайства бажано дешифрувати інформацію з поштового штемпелю на основі франкувального ключа у зоні локальної системи забезпечення платежів, наприклад, у поштових або вантажних центрах.

Для цього франкувальні ключі передають з центра передачі вартостей до центральної системи забезпечення платежів, бажано, автоматично.

Заміну бажано виконувати через сервер центральної системи забезпечення платежів (центральної ZinS). Центр передачі вартостей (Поштовий Пункт) можна не конфігурувати. Оскільки сервер не повинен знати центр передачі вартостей (локальних систем ZinS) і його відповідні криптосистеми, для зазначеного сервера важливим є лише центральний сервер ZinS.

Після нового генерування бажаного симетричного франкувального ключа його з захистом передають до центральної системи забезпечення платежів (опер.2 Фіг.1) і розподіляють звідси до криптосистем, розташованих у локальних системах забезпечення платежів (опер.3). Ці системи повертають результат імпортування (опер.4) і цим підтверджують успішне імпортування ключа. Результат компілюється центральною системою, підтверджується і передається до центра передачі вартостей (Поштового Пункту) (опер.5).

Після успішного імпортування ключа в усі криптосистеми він вивільняється у центрі передачі вартостей (Поштовому Пункті) і далі використовується для генерування нових поштових значень (опер.6).

Симетричні ключі є бажаними і є складовою частиною захисту від шахрайства поштових штемпелів, сформованих з використанням франкувального ключа, завдяки чому зазначені поштові штемпелі мають, наприклад, форму двомірних штрих-кодів. Отже, обмін цими ключами має бути захищений сильною криптографією. Для забезпечення цього важливо виконати такі вимоги:

- Конфіденційність вмісту: необхідно забезпечити неможливість зчитування переданих ключів третіми особами під час передачі. Крім того, необхідно забезпечити безпечне зберігання ключів. Це досягається застосуванням криптокарт WebSentry.

- Цілісність вмісту: неможливість підробки частин ключа третіми особами.

- Аутентифікація: обидва партнери, що мають зв'язок, мають бути впевнені, що ідентичність передавача/реципієнта є правильною. З точки зору реципієнта це означає, що ключ має бути генерований Поштовим Пунктом. З точки зору передавача має існувати гарантія, що приймає

ключ дозволено лише дійсним локальним криптосистемам ZinS.

У цьому симетричному варіанті обидва партнери мають однаковий ключ ТК. Центр передачі вартостей (Поштовий Пункт) використовує ключ ТК для шифрування ключа даних (КД), що має бути переданий, і потім передає зазначений ключ даних до сервера центральної системи забезпечення платежів (ZinS).

З центральної системи забезпечення платежів цей ключ розподіляється до локальних криптосистем ZinS локальної системи забезпечення платежів. Ці системи також мають доступ до ключа ТК і можуть щонайменше один раз дешифрувати ключ даних. Оскільки ключ ТК використовується для захисту ключа даних під час транспортування, його називають транспортним ключем.

Оскільки всі локальні системи забезпечення платежів приймають однакові дані, нема необхідності визначати окремий транспортний ключ між кожним локальним криптосервером і центром передачі вартостей (Поштовим Пунктом). З міркувань безпеки, однак, цей ключ необхідно оновлювати, як і ключ даних, з регулярним інтервалом. Але, оскільки текст, який шифрується цим ключем, не такий великий, як текст, що шифрується ключем даних, інтервал між обмінами може бути збільшений. Бажаний інтервал становить 1-2 роки.

Суттєвим компонентом такого рішення є заміна транспортних ключів, яке з міркувань безпеки не слід здійснювати через канал обміну ключем даних. Ця заміна не виконується вручну. Оскільки цю процедуру необхідно виконувати з регулярним інтервалом для декількох локальних систем забезпечення платежів, має бути застосований метод автоматичної заміни транспортних ключів.

Для цього стандарт ANSI X9.17 (Розпорядження ключами на основі симетричних методів) визначає інший рівень ключів, а саме, головний транспортний ключ (ГТК). Цей ключ встановлюють у криптокарті з спеціальними заходами безпеки і в подальшому він має застосовуватись зі значними інтервалами. У даному випадку в усіх системах встановлюють однаковий ГТК. Цей ключ шифрує транспортні ключі, які потім розподіляються і імпортується автоматично через той же канал. Розподілення здійснюється, як і для ключів даних. Ініціалізація індивідуальних систем або їх криптокарт розглядається нижче.

Для забезпечення цілісності повідомлення і аутентифікації передавача (Поштовий Пункт) формується матричний код (МК) для індивідуальних ключових повідомлень. Для генерування МК необхідно мати сигнатурний ключ (СК), який, як і ГТК, імпортується під час ініціалізації криптокарти. Ключем СК надається сигнатура для ключів даних. Цим ефективним криптографічним методом забезпечується конфіденційність інформації при передачі повідомлень у системі Intranet для Deutsche Post. Для шифрування і дешифрування повідомлень бажано застосовувати ключ Triple DES (довжина 168 байт). Хеш-значення бажано обчислювати, використовуючи алгоритм SHA-1.

При розподіленні і зберіганні ключів даних слід брати до уваги різні періоди дійсності у Поштовому Пункті і у криптосистемах систем забезпечення платежів. У будь-який момент у Поштовому Пункті мають бути доступними не більше двох ключів даних, а саме, один ключ з поточною дійсністю і ще один для шифрування щойно генерованих сум поштових зборів (КД<sub>нов</sub>).

У режимі "заміни" існуючого ключа ключем КД<sub>нов</sub> новий ключ передається до криптосистем систем забезпечення платежів. Після успішного імпортування цього ключа всіма криптосистемами локальних систем забезпечення платежів генерується повідомлення центральної системи ZinS про вивільнення і у цей момент КД<sub>нов</sub> використовується для шифрування нових сум поштових зборів у Поштовому Пункті. У цей момент старий ключ даних має бути стертий у Поштовому Пункті і більше не використовуватись для генерування нових сум поштових зборів.

Ситуація з криптосистемами локальних систем забезпечення платежів є іншою, оскільки завантажена сума поштових зборів може бути потрібною протягом невизначеного періоду, наприклад, трьох місяців, для формування поштових штемпелів необхідно мати декілька ключів даних для їх верифікації.

Коли справа доходить до розподілення ключів, слід брати до уваги долю ключів, які не можуть бути імпортовані у деякі криптосистеми і тому не можуть бути активовані у Поштовому Пункті. Можливо, ці ключі були імпортовані в інші криптосистеми і, у принципі, мають бути ураховані там під час майбутніх верифікацій.

Для забезпечення однакових умов, що дозволяє чітку верифікацію ключів і спрощує обслуговування системи, бажано забезпечити такі форми реалізації способу розподілення ключа:

а) Ключі даних передаються у шифрованій формі з однозначним ключем ID (індикатор фази ключа), невизначеним лічильником генерувань і кінцевою датою дійсності попереднього ключа даних. Лічильник генерувань використовується, щоб відрізнити багаторазові хибні спроби завантаження бажаних оновлених симетричних ключів даних (забезпечення безпеки транзакцій, див. g)).

б) Кожна передача ключа даних від центра передачі вартостей (Поштового Пункту)

має бути підтверджена центральною системою забезпечення платежів підтверджуючим повідомленням.

с) Якщо підтвердження є позитивним, це означає, що ключ даних був успішно встановлений в усіх локальних системах забезпечення платежів і може бути наданий клієнтам з використанням франкування РС. У цьому випадку лічильник генерувань інкрементується одиницею для передачі наступного ключа даних.

д) Якщо підтвердження є негативним, це означає, що ключ даних не був успішно встановлений в усіх локальних системах забезпечення платежів і не може бути наданий центром передачі вартостей (Поштовим Пунктом) системам клієнтів для формування поштових штемпелів, оскільки існує ризик масових хибних відправлень

цінної поштової кореспонденції. У цьому випадку лічильних генерувань не інкрементується і залишається з значенням попередньої передачі.

е) Якщо підтвердження відсутнє, центр передачі вартостей (Поштовий Пункт) не може почати подальшої передачі ключа до центральної системи забезпечення платежів (центральної ZinS) (такі спроби будуть ігноровані системою забезпечення платежів і мають бути заблоковані у центрі передачі вартостей).

ф) Якщо підтвердження системою забезпечення платежів залишається відсутнім протягом тривалого часу (довше зумовленої часової затримки), центр передачі вартостей (Поштовий Пункт) може ініціювати тривожне повідомлення через свої прямі або непрямі користувацькі інтерфейси.

г) Негайно після прийому ключа даних криптокартою, вона стирає всі ключі даних, які мають значення лічильника генерувань, яке відповідає останній передачі. Завдяки цьому у випадку багаторазових пов'язаних з помилками передач ключі, що раніше могли бути успішно завантажені у деякі криптокарти, зникають. Цим захищається передача ключів.

h) У криптокартах локальних систем забезпечення платежів багаторазово, бажано, регулярно, найкраще щоденно, ініціюється програма стирання ключів даних, в яких зникла потреба. Ключ даних стирається, коли кінцева дата, збережена для наступного ключа даних в атрибуті SKA\_END\_DATE є меншою за поточну системну дату.

i) Щодо кінцевої дати попереднього ключа, має бути запланований додатковий період (найкоротший з можливих), оскільки поштовий штемпель, сформований з сумою поштових зборів, яка втратила дійсність, вважається дійсним протягом ще двох днів після закінчення терміну дійсності при його перевірці у зоні локальної системи забезпечення платежів Крім того, має бути врахований час, що проходить між формуванням і вивільненням нового ключа даних.

Фіг.2 ілюструє варіанти застосування керування ключами згідно з винаходом і його використання у зоні системи забезпечення платежів. Показані також зони застосування.

Далі розглядаються варіанти застосування, зокрема, деталі описаного розпорядження ключами.

Застосування розглядається на прикладах використання криптокарт.

Перш за все розглядається ініціалізація криптокарт у зоні центра передачі вартостей (Поштового Пункту):

- Встановлення і конфігурування карти, завантаження "защитих" програм карти, якщо цього не зробив виробник. Функції "защитих" програм карти розширюються вбудованим кодом (власні процедури) і, для захисту узгоджені з PKCS 11 (Стандарти криптографії публічних ключів) процедури роботи з ключами (генерування, стирання тощо) мають бути заблоковані для користувача. Цим підвищується захист ключа.

- Визначення кластерів (з декількома криптокартами).

- Генерування і зберігання локального головного ключа (ЛГК), який підлягає розподіленню на щонайменше три компоненти, два з яких є особливо потрібними для відновлення щойно ініціалізованих криптосерверних карт. Бажано записувати кожний з компонентів у смарт-карту з захистом персонального коду, завдяки чому адміністратори безпеки отримують смарт-карти. Крім того, необхідно виготовити запасні смарт-карти. У подальшому ЛГК слугує як МТК.

- Користувацька адміністрація: стирання адміністратора безпеки за замовчування і визначення адміністратора безпеки з включенням правил аутентифікації реквізитів (на основі смарт-карти)

- Генерування початкового сигнатурного ключа (СК), шифрування (згортання) ГТК і зберігання як файлу. Копіювання цього файлу на дискету (доступ до файлів/дискет має лише адміністратор безпеки).

- Генерування першого ТК, створення належного ключового повідомлення у файлі і копіювання цього файлу на дискету (доступ до файлів/дискет має лише адміністратор безпеки).

Генерування МТК

Нові МТК бажано генерувати, як це описано нижче. Локальний головний ключ (ЛГК) належної конфірмаційної карти слугує як ГТК. З міркувань безпеки ЛГК розділяють на щонайменше три компоненти, з яких щонайменше два з них є необхідними для реімпортування.

Індивідуальні компоненти зберігаються у смарт-картах і захищені індивідуальними кодами (PIN), причому кожний адміністратор безпеки отримує смарт-карту і захищає її власним PIN. Зберігання смарт-карт у безпечному місці і секретність PIN виключає доступ до них сторонніх осіб.

Для користування ГТК бажано використовувати щонайменше два компоненти ГТК, які відповідають двом різним авторизованим картам, завдяки чому автоматично витримується принцип подвійного контролю.

ГТК має бути ключем типу Triple DES. Атрибут ID ключа складається з ідентифікатора типу і унікального номера.

Ідентифікатор типу: ТК

Унікальний номер: від 01 до 99

Довжина: фіксовані 4 байти у файлі SK\_CHAR.

Для забезпечення доступу до обміну ключами лише авторизованих осіб можна застосовувати різні механізми захисту. В описаному нижче втіленні використано сигнатурні ключі, що є особливо зручним, оскільки забезпечує особливо надійний захист від маніпуляцій.

Сигнатурний ключ (СК) забезпечує захищеність цілісності ключових повідомлень. Його можна також використовувати перед імпортом ключа для перевірки, чи є передавачем Поштовий Пункт Генерування СК може виконувати лише адміністратор безпеки, зареєстрований через смарт-карту. Це має бути ключ TDES з значенням атрибутом захисту "чутливий" TRUE і атрибутом "ви-

добування" FALSE для запобігання виявлення тексту ключа назовні смарт-карти.

Атрибут ID ключа складається з ідентифікатора типу і унікального номера.

Ідентифікатор типу: CK

Унікальний номер: від 01 до 99

Довжина: фіксовані 4 байти у файлі CK\_CHAR.

Для експортування ключа він має бути згорнутий ключем ГТК і збережений як файл на дискеті, яку необхідно зберігати у безпечному місці і використовувати для ініціалізації криптосерверних карт. Цілісність файлу ключа забезпечується програмою обробки, що знаходиться у карті і використовується для згортання.

Наведена нижче таблиця містить бажані атрибути CK ключа ТК.

Атрибути CK транспортного ключа	
Атрибут ключа	Значення
CKA_ID	СК+ порядковий унікальний номер (від KS01 до KS99) для унікального призначення ключа
CKA_LABEL	Не заповнюється; відповідає схемному значенню за замовчування у момент генерування
CKA_START_DATE	Дата, від якої ключ вважається дійсним, у форматі PKCS 11 (встановлюється адміністратором безпеки)
CKA_END_DATE	Дата, після якої попередній ключ має бути стертий
CKA_SENSITIVE	TRUE
CKA_EXTRACTABLE	FALSE
CKA_ENCRYPT	TRUE
CKA_DECRYPT	TRUE

Рандомізоване число в атрибуті LABEL слугує для підтвердження успішного імпортування у криптосерверні карти. Для цього випадкового числа формується хеш-значення (наприклад, засобами SHA-1) і використовується для підтвердження успішного імпортування, а також вивільнення ключа.

Атрибути CKA\_ID і CKA\_LABEL заповнюються як CK\_CHAR. Всі подальші атрибути визначаються як типи через файл pkcs11.h.

Сигнатурний ключ шифрується ключем ГТК (=ЛГК, механізм CKM\_KEY\_WRAP\_WEBSENTRY) і завантажується у схему на місці як ЛГК.

Нижче розглядається генерування ТК.

У цьому варіанті застосування генерується ТК, який включає належне ключове повідомлення. Бажано, щоб конфігурація модуля генерування ключа забезпечувала генерування ТК і ініціалізацію належного ключового повідомлення лише адміністратором безпеки. Інтервал між замінами має становити 1-2 роки

ТК є ключем типу TDES з такими атрибутами:

Атрибути ТК	
Атрибут ключа	Значення
CKA_ID	ТК + порядковий унікальний номер (від KT01 до KT99) для унікального призначення ключа
CKA_LABEL	Випадкове значення довжиною 64 байти, яке генерується методом C GenerateRandom PKCS 11
CKA_START_DATE	Дата, від якої ключ вважається дійсним, у форматі PKCS 11 (встановлюється адміністратором безпеки)
CKA_END_DATE	Дата, після якої попередній ключ має бути стертий
CKA_SENSITIVE	TRUE
CKA_EXTRACTABLE	FALSE
CKA_ENCRYPT	TRUE
CKA_DECRYPT	TRUE

Рандомізоване число в атрибуті LABEL слугує для підтвердження успішного імпортування у криптосерверні карти. Для цього випадкового числа формується хеш-значення (наприклад, засобами SHA-1) і використовується для підтвердження успішного імпортування, а також вивільнення ключа.

Атрибути CKA\_ID і CKA\_LABEL заповнюються як CK\_CHAR. Всі подальші атрибути визначаються як типи через файл pkcs11.h.

ТК шифрується ключем ГТК (=ЛГК, механізм CKM\_KEY\_WRAP\_WEBSENTRY) і для передачі від центра передачі вартостей (Поштового Пункту) до центральної системи узбепечення платежів формується така структура:

П.п.	Довжина	№№ байт	Призначення	Опис
1	2	f1-f4	MsgType	Ідентифікатор ключового повідомлення, константа "КТ"
2	1	f3	Version	Версія структури повідомлення, 1 байт, що починається з значення "01"
3	4	f4-f8	KeyID	Ідентифікатор ТК прямим текстом
4	4	f9-f12	SigKeyID	Ідентифікатор ключа, що використовується для сигнатури
5	n-13	f13-f <sub>n1</sub>	TranspKeyEncrypt	Результат шифрування (згорання) ТК
6	24	f <sub>n+1</sub> -f <sub>n+24</sub>	MAC	МК для ключового повідомлення; формується як хеш-значення SHA-1 для полів 1-5 повідомлення, причому це значення шифрується сигнатурним ключем (механізм CKM_DES3_CBC_PAD, IV заповнюється нулями, ID - див. поле f3.

Після подальшого розподілення це повідомлення ТК передається до центрального сервера ZinS. Інтерфейс - типу Session Bean, це обслуговування виявляють з використанням сервісу найменувань (Java Naming and Directory Interface - JNDI). Спосіб транспортування має урахувати наведений вище блок даних.

Це повідомлення зберігається у Поштовому Пункті як файл, і адміністратор безпеки може потім зберігати його на одній або декількох дискетах у безпечному місці. У подальшому ці дискети використовуються для ініціалізації нових криптосерверних карт.

Далі розглядається бажане втілення процедури вивільнення ТК

Конфігурація центра передачі вартостей дозволяє вивільняти ТК. Для цього існує інтерфейс, через який центр передачі вартостей може вивільняти цей ТК, як тільки він був успішно розподілений і імпортований в усі системи забезпечення платежів (криптосистеми ZinS). Лише через вивільнення можна використовувати належний ТК для шифрування ключів даних.

Інтерфейс реалізований як Session Bean. Це обслуговування виявляють з використанням сервісу найменувань. Бажана структура даних для процедури вивільнення має такі параметри:

П.п.	Довжина	№№ байт	Призначення	Опис
1	4	f1-f4	KeyID	Ідентифікатор ТК прямим текстом.
2	201	f4-f24	RandomHash	Хеш-значення SHA-1 рандомізованого числа переноситься як атрибут LABEL ТК.
3	1	f25	State	Результат шифрування: TRUE= є необхідною обробка: вивільнення можливе, FALSE= обробка не була успішною.
4	128	f26-f153	Message	Детальне повідомлення про причину помилки або детальне повідомлення про успіх.

Аутентифікація призначення ключа системи забезпечення платежів (система KeyManagement ZinS) vis-à-vis PP KeyManagement system виконується з використанням параметра 2, тобто хеш-значення, отриманого методом SHA-1 з атрибута LABEL ТК. Цей атрибут заповнюють випадковим значенням під час генерування ТК. Оскільки ТК і всі його атрибути шифруються під час передачі, лише криптосервер ZinS може обчислити правильне хеш-значення.

Якщо передане хеш-значення є ідентичним хеш-значенню ТК, обчисленого для атрибута LABEL, і це хеш-значення знаходиться у модулі WebSentry Поштового Центра, а переданий статус обробки є TRUE, то ТК активується.

Це означає, що у подальшому повідомлення ключа даних шифруватимуться ТК.

Цей варіант застосування може існувати, якщо обробка є хибною (переданий статус - FALSE). У цьому випадку статус для цього генерування ключа і його розподілення протоколюється і відповідний ТК стирається.

Далі розглядається приклад генерування нових ключів даних.

У даному випадку генерується ключ даних з належним ключовим повідомленням. Бажано, щоб система призначення ключів мала конфігурацію, яка дозволяє ініціювання цієї дії лише адміністратору безпеки. Це необхідно проводити кожні 3 місяці. Якщо ключ даних є в обігу і для нього від центральної системи забезпечення платежів (центральної системи ZinS) ще не було отримано зворотного зв'язку (вивільнення або помилка), генерування нових ключів даних блокується до отримання зворотного зв'язку.

Ключ даних використовується для шифрування вмісту певного матричного коду і для забезпечення того, щоб жоден поштовий штампель не був сформований, якщо не було зроблено жодного платежу провайдеру поштових послуг. Цей ключ даних слугує для верифікації цифрового поштового штампеля у криптосерверах ZinS. Ключі даних також є ключами TDES, що генеруються з використанням функції C\_GenerateKey, і мають такі атрибути:

Атрибути ключа даних	
Атрибут ключа	Значення
СКА_ID	Порядковий унікальний номер (наприклад, 01) для унікального призначення ключа (без префіксу); відповідає індикатору фази ключа, який закодовано у поштовому штемпелі і у PostalID.
СКА_LABEL	1-й байт: лічильник генерувань для спрощеного стирання ключів у криптосистемах ZinS, які не можна активувати. Байти 2-65: випадкове значення довжиною 64 байти, яке генерується методом C_GenerateRandom PKCS 11.
СКА_START_DATE	Дата, від якої ключ вважається дійсним, у форматі PKCS 11 (встановлюється адміністратором безпеки)
СКА_END_DATE	Вказує не дату кінця терміну дійсності ключа, а дату кінця дійсності попередника тобто дату, після якої попередній ключ має бути стертий.
СКА_SENSITIVE	TRUE
СКА_EXTRACTABLE	FALSE
СКА_ENCRYPT	TRUE
СКА_DECRYPT	TRUE

Значення для атрибута СКА\_ID встановлюється системою. СКА\_ID завжди інкрементується одиницею, а лічильник генерувань інкрементується лише у випадках успішного вивільнення. Атрибути СКА\_ID і СКА\_LABEL заповнюються як СК\_CHAR. Типи подальших атрибутів визначаються через файл pkcs.h.

Рандомізоване число в атрибуті LABEL слугує для підтвердження успішного імпортування у криптосерверні карти. Для цього числа формується хеш-значення (засобами SHA-1) і воно ви-

користовується для підтвердження успішного імпортування і вивільнення ТК.

Генерування повідомлення для заміни ключа даних є дещо складнішим і складається з таких дій:

1. Ключ даних шифрують ГТК (=ПГК, механізм СКМ\_KEY\_WRAP\_WEBSENTRY). Це дає ту перевагу, що всі атрибути (між іншими, СКА\_EXTRACTABLE=FALSE) шифруються і їх значення відновлюються під час дешифрування. Цією байтовою послідовністю репрезентується проміжне повідомлення такої структури:

Структура проміжного повідомлення для ключа даних				
П.п.	Довжина	№№ байт	Призначення	Опис
1	n	f1-n	DateKeyEncrypt	Результат шифрування (згортання) ключа даних спеціальним механізмом WebSentry (ГТК)

2. Проміжне повідомлення у свою чергу шифрується активним ТК з використанням механізму СКМ\_DES3\_CBC\_PAD (вектор IV ініціалізації заповнюється нулями).

3. Формується повідомлення, що підлягає розподіленню і має таку структуру:

Структура повідомлення для ключа даних, призначеного для розподілення				
П.п.	Довжина	№№ байт	Призначення	Опис
1	2	f1-f4	MsgType	Ідентифікатор ключового повідомлення, константа "КТ"
2	1	f3	Version	Версія структури повідомлення, 1 байт, що починається з значення "01"
3	1	f4	Key ID	Ідентифікатор ТК у прямому тексті
4	2	f5-f7	KeyGeneration	Лічильник генерувань прямим текстом, наприклад, "01"
5	4	f8-f12	SigKeyID	Ідентифікатор ключа, застосованого до сигнатури
6	4	f12-f16	KTID	Ідентифікатор ТК, застосованого для шифрування проміжного повідомлення
7	n	f16-f <sub>16</sub>	HelperMsgEncrypt	Результат шифрування проміжного повідомлення
8	24	f <sub>n1+1</sub> -f <sub>n1+24</sub>	MAC	MAC для проміжного повідомлення формується як хеш-значення SHA-1 для поля 7 повідомлення і це хеш-значення шифрується сигнатурним ключем (механізм СКМ_DES3_CBC_PAD, IV заповнюється нулями, ID - див. поле 5).

4. Далі повідомлення ключа даних передається для розподілення до центрального сервера ZinS. Воно зберігається адміністратором безпеки на одній або декількох дискетах у безпечному місці. У подальшому ці дискети використовуються для ініціалізації нових крипто-серверних карт.

Перевагою подвійного шифрування вмісту повідомлення є зменшення шифру, що має бути переданий до ГТК через Інtranет, і ускладнення криптоаналізу цього ключа.

Для вивільнення ключа даних застосовується інтерфейс і протокольний механізм, які забезпе-

чують протоколювання вивільнення ключа. Центральна система забезпечення платежів бажано конфігурувати таким чином, щоб ключі даних вивільнялися лише після успішного розподілення і успішного імпортування цього ключа у криптосистеми локальних систем забезпечення платежів. Лише завдяки цьому вивільненню належний ключ даних у подальшому успішно використовується для шифрування крипторядків, що мають бути введені у поштовий штампель, після чого ідентифікаційна інформація KeyID ключа

кодується у ідентифікаційну інформацію (PostageID) поштових платежів.

Інтерфейс, реалізований через CWMS (комп'ютеризована робоча система менеджменту), діє між центральною системою забезпечення платежів (центральна ZinS) і локальними системами забезпечення платежів (локальні ZinS). Центр передачі вартостей (Поштовий Пункт) 33 приймає інформацію через відповідний вхід, Структура даних для процедури вивільнення має вигляд:

П.п.	Довжина	№№ байт	Призначення	Опис
1	1	f1	Key ID	Ідентифікатор ТК прямим текстом.
2	20	f2-f22	RandomHash	Хеш-значення SHA-1 рандомізованого числа переноситься як атрибут LABEL ключа даних.
3	1	f23	State	Результат шифрування: TRUE - успішна обробка: вивільнення можливе, FALSE - обробка не була успішною.
4	128	f23-f151	Message	Детальне повідомлення про причину помилки або детальне повідомлення про успіх.

Аутентифікація системи призначення ключів системи забезпечення платежів відносно системи системи призначення ключів центра передачі вартостей (Поштового Пункту) виконується через параметр 2. Бажано, щоб це було хеш-значення, яке з застосуванням метода SHA-1 формується з атрибута LABEL ключа даних. Атрибут LABEL заповнюється рандомізованим числом під час генерування ключа даних. Оскільки ключ даних і всі його атрибути шифруються під час передачі, лише криптосервер системи забезпечення платежів може обчислити правильне хеш-значення.

Якщо передане хеш-значення є ідентичним хеш-значенню ключа даних, обчисленого для атрибута LABEL, і це хеш-значення знаходиться у модулі WebSentry Поштового Центра, а переданий статус обробки є TRUE, то ключ даних активується. Це означає, що у подальшому крипторядки для поштових штампелів/поштових платежів шифруватимуться цим ключем даних.

Цей варіант застосування може існувати, якщо обробка є хибною (переданий статус - FALSE). У цьому випадку статус для цього генерування ключа і його розподілення протоколюється і відповідний ключ даних на карті стирається, а лічильник генерувань не інкрементується.

Бажано, щоб система призначення ключів мала статусну пам'ять для зберігання виконаних генерувань ключа. Поки не одержано зворотного зв'язку від центральної системи забезпечення платежів (центральної системи ZinS) про виконане розподілення ключа, статус репрезентується як відкритий. Після отримання успішного зворотного зв'язку і розподілення, ключ призначається як активований. У випадку помилки надається передане повідомлення про статус. При наявності помилок або тривалій відсутності повідомлення про вивільнення виконується програма аналізу помилки.

Бажано виконувати архівування ключів. Бажаний варіант архівування ключів у зоні центра передачі вартостей (Поштового Пункту) розглядається нижче. Для захисту ключів архіватор мо-

же ініціювати функцію архівування, яка шифрує всі ключі (за винятком ГТК) ключем ГТК (механізм SKM\_KEY\_WRAP\_WEBSENTRY) і повертає їх. Ці ключі слід зберігати у базі даних або у захищеній зоні файлової системи.

Після успішного архівування стираються всі ключі, для яких початковий термін дієвості минув більш, ніж 6 місяців тому, і які не є активними.

Ключі відновлюються, зокрема у зоні центра передачі вартостей (Поштового Пункту), тобто архівовані дані ключа дешифруються (розгортаються) знову і зберігаються у карті. Для цього знову використовується механізм SKM\_KEY\_WRAP\_WEBSENTRY.

Після дешифрування ключа ключ з такою ж ідентифікаційною інформацією KeyID, який вже є у карті, стирається. Завдяки спеціальним заходам безпеки у карті WebSentry і розподіленню між декількома смарт-картами, ГТК надійно захищений проти втручання.

Якщо необхідно замінити ГТК, то, як у випадку ініціалізації криптокарти (ПП), мають бути генеровані нові сигнатурні ключі, транспортні ключі і ключі даних.

Попередній ГТК залишається у карті як так званий "бездіяльний ключ" і має бути стертий адміністратором безпеки.

Далі розглядаються бажані варіанти застосування системи призначення ключів в усіх згідно з одним з аспектів системи забезпечення платежів. Це дає особливі переваги, коли індивідуальні компоненти застосовуються, бажано, у зоні локальної системи забезпечення платежів або центральної системи забезпечення платежів. Використання інших систем забезпечення платежів також є припустимим.

Першим варіантом застосування є ініціалізація криптокарти у зоні системи забезпечення платежів.

Для ініціалізації криптокарти мають бути виконані дії, більшість яких можуть бути здійснені з використанням інструмента менеджменту WebSentryManager:

- Встановлення і конфігурування карти, завантаження вбудованих програм, якщо цього не зробив виробник. Вбудовані програми містять Embedded Code (вбудований код власних програм). Ця функція вбудована у WebSentryManager.

- Визначення кластерів (з декількома криптокартами) (WebSentryManager).

- Імпортуюння ГТК, див. окремий варіант застосування (функції виконує WebSentryManager).

- Користувачка адміністрація: стирання адміністратора безпеки за замовчування і визначення адміністратора безпеки з включенням правил аутентифікації реквізитів (на основі смарт-карти).

- Генерування двох "нормальних" користувачів (один для верифікації ключа, один для імпортування ключа), які авторизують себе через задалегідь визначений PIN. Тут функції також виконує WebSentryManager.

- Імпортуюння СК (див. див. окремий варіант застосування).

- Імпортуюння ТК (див. див. окремий варіант застосування).

- Як варіант, імпортування ключів даних (див. варіант їх застосування).

Ключі мають імпортуватись у наведеній вище послідовності. Карту можна ініціювати у центральному місці, причому має бути ініціалізований комп'ютер повної криптосистеми, оскільки карта WebSentry стирає внутрішню пам'ять, як тільки карту виймають з щілини PCI.

Бажано, щоб ГТК міг бути імпортований лише щонайменше двома адміністраторами безпеки, які локально ідентифікують себе для криптосистеми за допомогою зчитувача з смарт-карт і відповідної смарт-карти. Важливість ГТК вимагає, щоб цей ключ міг бути імпортований у картку лише під подвійним контролем. Це означає, що у варіанті застосування "головний транспортний ключ" потрібно мати щонайменше дві захищені персональним кодом PIN смарт-карти.

Оскільки ГТК може бути завантажений у карту лише з використанням обох смарт-карт, а використання ключа захищене PIN, це гарантує, що цей ключ може бути встановлений у карті лише під подвійним контролем. Цим забезпечується дуже високий рівень захисту від доступу до ключа.

Ця функція забезпечується інструментом WebSentryManager, який дозволяє завантажувати так званий локальний головний ключ (ЛГК відповідає ГТК у цьому варіанті) з використанням смарт-карти і зберігати їх у захищеній зоні карт.

Особливо зручно розподіляти ЛГК між трьома смарт-картами, причому для імпортування ЛГК у криптообладнання необхідно мати всі три смарт-карти, а в імпортуванні ГТК мають брати участь три адміністратори безпеки.

Сигнатурний ключ забезпечує цілісність ключових повідомлень; його можна також використовувати перед імпортуванням ключа для перевірки, чи є передавачем центр передачі вартостей (Поштовий Пункт). Генерування СК можна здійснювати різними шляхами, але бажаними є ілюст-

ровані наведеними тут прикладами.. Належне ключове повідомлення зберігається адміністратором безпеки на дискеті і, у даному варіанті, імпортується у криптокарту системи убезпечення платежів, яка підлягає ініціалізації.

Для імпортування ключа ключове повідомлення дешифрується за допомогою ГТК (функція C\_Unwrap PKCS 11, механізм SKM\_KEY\_WRAP\_WEBSENTRY). Ця програма автоматично перевіряє цілісність цього повідомлення. Якщо ключ з таким KeyID вже існує, його стирають.

Для подальшого транспортування транспортних ключових повідомлень сервер системи убезпечення платежів надає інтерфейс, через який можуть бути ініційовані розподілення і подальше імпортування у криптосистеми індивідуальних систем убезпечення платежів.

Інтерфейс реалізують як сервіс RMI, який викликають засобами сервісу найменувань (JNDI). Розподілення виконується через CWMS (комп'ютеризована робоча система менеджменту), яка розподіляє список P/N.

Якщо створено нову функцію розподілення, ключове повідомлення пересилається до всіх поточно зареєстрованих криптосистем з складанням протоколу у кожному випадку Розпорядження криптосистемами здійснюється з застосуванням системи убезпечення платежів.

Прийом нових ключових повідомлень в індивідуальних криптосистемах перевіряється засобом ImportController з регулярним інтервалом (як функція механізму розподілення). Після прийому нового повідомлення автоматично ініціюється функція застосування "імпортування ТК" Значення, яке повертає ця процедура, перевіряється. У випадку негативного зворотного зв'язку спроба імпортування повторюється до трьох разів.

Якщо після трьох спроб імпортування не проходить, до центральної системи ZinS надсилається протокольне повідомлення про неуспіх (як функція механізму транспортування). У випадку успішного імпортування надсилається позитивне протокольне повідомлення.

Протокольні повідомлення обробляються функцією "обробка ключа протоколу". Відповідним чином ініціюється вивільнення ТК.

Імпортуюння ТК виконується адміністратором безпеки, який ініціалізує криптосистему на місці, або це імпортування ініціюється автоматично функцією ImportController розподілення ключів, коли ImportController отримує нове транспортне ключове повідомлення. Бажана процедура імпортування ключа включає такі операції:

1. Реєстрація у карті через ID і PIN користувача KeyImport.

2. Для полів 1-5 транспортного ключового повідомлення формується хеш-значення методом SHA-1.

3. Підтверджується СК, який відповідає KeyID поля 4.

4. Цей ключ використовується для шифрування хеш-значення, сформованого у п.2 (механізм SKM\_DES3\_CBC\_PAD, вектор IV ініціалізації заповнюється нулями), яке порівнюється з полем

6. Якщо значення збігаються, цілісність підтверджується, і це означає, повідомлення було створене франкувальною системою PC.

5. Вміст поля 5 повідомлення дешифрується ключем ГТК (метод C\_UnwrapKey, механізм SKM\_KEY\_WRAP\_WEBSENTRY). В результаті належний об'єкт ТК генерується автоматично і зберігається у карті. Крім того, цей механізм знову неявно перевіряє цілісність ключа і джерело.

6. Якщо ключ з таким же KeyID вже існує, він стирається.

7. Для атрибута LABEL щойно імпортованого ключа методом SHA-1 формується хеш-значення, яке потім повертається разом з KeyID і позитивним повідомленням як повернуте значення.

8. Кінець користувацького сеансу.

9. З повернутих значень формується протокольне повідомлення, яке надсилається до центральної системи ZinS.

10. Будь-яка неуспішна спроба, збережена для цього KeyID, повторюється (див. варіант, описаний нижче).

У випадку, коли не спрацює одна програм або перевірка MAC, подальша обробка припиняється і повертається значення, яке містить KeyID, код помилки і повідомлення про помилку. Щодо KeyID, збережена кількість невдалих спроб підвищується на 1. Коли ця кількість досягає 3, на основі останнього переданого повернутого значення формується протокольне повідомлення і надсилається до центральної системи забезпечення платежів.

У випадку ручного ініціювання результат імпортування виводиться на монітор адміністратора безпеки.

Бажано, щоб операції 2-7 виконувались програмами (вбудованим кодом) карти. Це підвищує ефективність і захист від проникнення.

Для подальшого транспортування повідомлень ключа даних сервер центральної системи забезпечення платежів надає ще один інтерфейс, через який здійснюється розподілення і подальше імпортування ключів даних в індивідуальні криптосистеми локальних систем забезпечення платежів.

Інтерфейс реалізовано як Sessio Bean; це обслуговування викликають з використанням сервісу найменувань (Java Naming and Directory Interface - JNDI).

Розподілення виконується через CWMS (комп'ютеризована робоча система менеджменту), яка розподіляє список P/N.

При створенні нової функції розподілення ключове повідомлення пересилається до всіх поточно зареєстрованих криптосистем з складанням протоколу у кожному випадку. Розпорядження криптосистемами здійснюється з застосуванням системи забезпечення платежів. Якщо ключ даних вже є у обігу, для якого від центра передачі вартостей (Поштового Пункту), розподілення нових ключів даних блокується до отримання зворотного зв'язку.

Прийом нових ключових повідомлень перевіряються в індивідуальних криптосистемах за до-

помогою ImportController з регулярним інтервалом (як функція механізму розподілення) Після прийому нового повідомлення автоматично ініціюється функція застосування "імпортування ТК". Значення, яке повертає ця процедура, перевіряється. У випадку негативного зворотного зв'язку спроба імпортування повторюється до трьох разів.

Якщо після трьох спроб імпортування не проходить, до центральної системи ZinS надсилається протокольне повідомлення про неуспіх (як функція механізму транспортування). У випадку успішного імпортування надсилається позитивне протокольне повідомлення.

Протокольні повідомлення обробляються функцією "обробка ключа протоколу". Відповідним чином ініціюється вивільнення ТК.

Імпортування ТК виконується адміністратором безпеки, який ініціалізує криптосистему на місці, або це імпортування ініціюється автоматично функцією ImportController розподілення ключів, коли ImportController отримує нове повідомлення ключа даних. Бажана процедура імпортування ключа включає такі операції:

1. Реєстрація у карті через ID і PIN користувача KeyImport.

2. Для полів 1-7 транспортного ключового повідомлення формується хеш-значення методом SHA-1.

3. Підтверджується СК, який відповідає KeyID поля 5.

4. Цей ключ використовується для шифрування хеш-значення, сформованого у п.2 (механізм SKM\_DES3\_CBC\_PAD, вектор IV ініціалізації заповнюється нулями), яке порівнюється з полем 8.

8. Якщо значення збігаються, цілісність підтверджується, і це означає, повідомлення було створене франкувальною системою PC.

5. Визначається ТК, якому належить KeyID, визначений у полі 6.

6. Вміст поля 7 повідомлення дешифрується ключем, визначеним у п.5 (метод C\_Decrypt, механізм SKM\_DES3\_CBC\_PAD, вектор IV ініціалізації заповнюється нулями), яке порівнюється з полем 8. Результатом дешифрування є проміжне повідомлення.

7. Вміст поля 1 проміжного повідомлення дешифрується ключем ГТК (метод C\_UnwrapKey, механізм SKM\_KEY\_WRAP\_WEBSENTRY). В результаті належний об'єкт ТК генерується автоматично і зберігається у карті. Крім того, цей механізм знову неявно перевіряє цілісність ключа і джерело.

8. Якщо ключ з таким же KeyID вже існує, він стирається.

9. Всі ключі даних у криптокарті зчитуються і перевіряються на ідентичність значення лічильника генерувань в атрибуті LABEL (байт 1) цьому значенню щойно прийнятого ключа. Якщо так, ці ключі видаляються з карти. Ці ключі є тими, що не були вивільнені у центрі передачі вартостей (Поштовому Пункті) внаслідок помилок імпортування у криптосистеми іншої локальної системи забезпечення платежів.

10. Для байтів 2-65 атрибута LABEL щойно імпортованого ключа формується хеш-значення (з використанням SHA-1) і повертається разом з KeyID і позитивним повідомленням.

11. Кінець користувацького сеансу.

12. З повернутих значень формується протокольне повідомлення, яке надсилається до центральної системи ZinS.

10. Будь-яка неуспішна спроба, збережена для цього KeyID, повторюється (див. варіант, описаний нижче).

У випадку, коли не спрацює одна програм або перевірка MAC, подальша обробка припиняється і повертається значення, яке містить KeyID, код помилки і повідомлення про помилку. Щодо KeyID, збережена кількість невдалих спроб підвищується на 1. Коли ця кількість досягає 3, на основі останнього переданого повернутого значення формується протокольне повідомлення і надсилається до центральної системи забезпечення платежів (центральної ZinS).

У випадку ручного ініціювання результат імпортування виводиться на монітор адміністратора безпеки.

Бажано, щоб операції 2-10 виконувались програмами (вбудованим кодом) карти. Це підвищує ефективність і захист від несанкціонованого доступу (особливо доступу до вектора IV ініціалізації і ГТК).

Очищення ключів даних виконується, бажано, регулярно, у якнайбільшій кількості криптосистем, бажано в усіх, з метою стирання тих ключів, що вже не є потрібними.

Процедура очищення ключів даних:

1. Всі ключі даних у карті перевіряються і упорядковуються у порядку зростання їх ідентифікаторів (ID), тобто атрибута SCA\_ID.

2. Для кожного ключа з цього списку виконується така процедура перевірки:

I. Визначається наступний ключ (з наступним більшим ID).

II. Якщо такий ключ є, відбувається перевірка того, що:

1. атрибут SCA\_END\_DATE наступного ключа, який вказує дату закінчення дійсності, є меншим з поточною датою, і якщо так, то ключу з списку, що у цей момент обробляється, стирається;

2. лічильник генерувань наступного ключа (байт 1 атрибута SCA\_LABEL) є ідентичним лічильнику генерувань ключа, що обробляється у цей момент; якщо це так, ключ, що обробляється у цей момент, стирається.

Обробку ключа бажано протоколювати у сервері центральної системи забезпечення платежів (центральному сервері ZinS). Ключі, що протокуються у даному випадку, є ТК і ключем даних.

Для кожної операції розподілення сформований протокол вказує, до якої з активних криптосистем було надіслано ключове повідомлення. Для кожної системи і кожної операції розподілення формується окремий запис з початковим статусом "надіслано".

Після кожної успішної і неуспішної обробки ключа індивідуальні криптосистеми генерують протокольне повідомлення і надсилають його до

центральної системи забезпечення платежів (центральної системи ZinS). Це розподілення здійснюється з застосуванням черг JMS або з реплікацією бази даних.

У зоні центральної системи забезпечення платежів після прийому повідомлень вони використовуються для оновлення зазначених вище протокольних записів. Для цього зберігається статус "успішна обробка" або код помилки і повідомлення.

Після обробки протокольних повідомлень здійснюється перевірка, чи існують операції розподілення, успішно введені всіма криптосистемами. Якщо так, ініціюється вивільнення кожного відповідного ключа, зокрема, у зоні центра передачі вартостей (Поштового Пункту). Як тільки система сповіщає про помилку, у зоні центра передачі вартостей (Поштового Пункту) формується відповідне повідомлення з негативним статусом.

Той факт, що було ініційоване вивільнення ключа відзначається у примітці разом з операцією розподілення і тому для даної операції додаткове вивільнення не здійснюється. Але доки ця примітка не введена, повторюються (з регулярним інтервалом) спроби контакту з службою вивільнення.

Особлива ситуація створюється, коли після визначеного періоду часу, бажано, через декілька днів, наприклад, три дні, не отримується зворотний зв'язок від усіх криптосистем. У цьому випадку до центра передачі вартостей (Поштового Пункту) надсилається негативне повідомлення про вивільнення.

Бажано, щоб система призначення ключів мала користувацький інтерфейс, який дозволяє адміністратору перевіряти статус операції розподілення ключа. Для кожної операції розподілення на індикацію виводяться такі дані:

- кількість систем до яких було надіслано повідомлення про розподілення,
- кількість систем, які доповіли про успішну обробку,
- кількість систем, які ще не доповіли про результат обробки,
- кількість систем, які доповіли про неуспішну обробку.

Крім того, може бути складений перелік поточних статусів всіх цих систем.

В іншому варіанті можна локально виводити на індикацію ключі, введені у належні карти.

Доцільно архівувати у зоні центральної системи забезпечення платежів всі ключі, для яких були ініційовані операції розподілення. Бажано не проводити архівування у локальних системах. У них ключі зберігаються у енергонезалежній пам'яті карти. Архівуються лише ті ключові повідомлення, які були також вивільнені.

Відновлення ключів ТК і ключів даних можна ініціювати централізовано для конкретної криптосистеми. У цьому випадку поточні ключі виявляються в архіві і надсилаються до цієї криптосистеми. Для цього формуються також протокольні повідомлення. У такому варіанті розподілення

ключів відсутнім є лише повідомлення про вивільнення.

У випадку втрати ГТК відповідну криптосистему надсилають до адміністраторів безпеки для нової ініціалізації або адміністратори безпеки мають ініціалізувати кожну систему на місці.

ГТК надійно захищений проти втручання спеціальними заходами безпеки карти Web-Sentry і розподіленням у декількох смарт-картах, а також багатостадійною системою ключів.

Якщо має бути проведена заміна ГТК, необхідно генерувати новий ГТК, транспортні ключі і ключі даних.

Після цього вони мають бути імпортовані в усі криптосистеми локальних систем убезпечення платежів. Це потребує більше операцій, оскільки необхідно або транспортувати всі криптосистеми до місця центральної адміністрації і назад, або адміністратори безпеки мають відвідати всі місця локальних систем убезпечення платежів. Тому використання механізму захисту для ГТК згідно з винаходом є особливо зручним. Попередній ГТК запишається у карті як так званий "бездіяльний ГТК" і має бути стертий адміністратором безпеки.

Робота з ключем і дешифрування запрограмовані у криптокарті вбудованим кодом. Цим досягається не лише кращий захист, але й підвищення ефективності криптосистеми.

Бажано, щоб криптокарти містили такі стандартні функції PKCS 11:

```
C_CloseSession
C_GetSlotList
C_Init
C_Initialize
C_Login
C_Logout
C_OpenSession
```

разом з розширеннями. Крім того, функції, що зберігаються постійно, не повинні мати будь-яких подальших розширень третіми особами, а необхідні розширення ексклюзивно інтегруються як функції для криптокарт системи убезпечення платежів.

Для позначення вбудованих операцій DLL PKCS 11 їм надано префікс CE\_ (Crypto Extension). Кожний вбудований метод повертає значення типу CK\_RV, який визначається як фіксований include-файл pkcs11.h. Зручним є те, що під час використання вбудованих функцій повертаються додаткові коди помилок і вносяться у файл заголовка C++ для інтегрування. Це надає ту перевагу, що викликом вбудованої функції викликаються різні методи Pkcs11, включені у схему. Іншою перевагою є застосування нового порядку роботи з ключами, визначеного програмами криптокарт.

Приклад синтаксису цього способу:

CK\_RV= CE\_MethodName (список параметрів)

У цьому списку параметрів параметри, що мають бути заповнені результатом, передаються через виклик посиланням. Ім'я методу формується з комбінацій значущих слів, які дають уявлення про метод.

Вибір слів здійснюється таким чином, щоб створити асоціацію з вмістом, наприклад, використовуючи англійську технічну термінологію.

Введення двох типів нумерації слугує для верифікації функцій різних вбудованих методів. Типи нумерації типів ключів і атрибутів ключів є різними.

CE\_EnumKey= {CE\_KT, CE\_DT, CE\_SE, CE\_KA}

CE\_KA займає особливу позицію. Це не ключ, а, скоріше, набір всіх ключів. Якщо вказаний цей KeyElement, то метод виконує функцію, що стосується всіх ключів у карті.

CE\_EnumKeyAttribute= (CF\_ID, CE\_LABEL, CE\_STARTDATE, CE\_ENDDATE)

Необхідні значення необхідно внести у файл pkcs.h. Визначені розширення можна розмістити в окремому заголовку, включеному у pkcs.h. Для реалізації вбудованих методів можуть бути використані різні механізми.

У криптографічному середовищі визначається метод, який виконує всі необхідні верифікації наявними методами PKCS 11.

CK\_RV CE\_Decrypt (сеанс CK\_SESSION\_HANDLE)

повідомлення CK\_BYTE [], поштовий ідентифікатор CK\_BYTE\*

CK\_BBOOL bOk

Опис функції:

Вбудований криптографічний метод приймає 57-байтову дату у параметричному повідомленні, яка відповідає матричному коду поштового штемпелю. У наведеному далі описі відлік починається з одиниці.

Операція 1: формування вектора IV ініціалізації, в якому перші два байти заповнюються нулями, а байти f6-f10, f14 додаються до матричного коду.

Операція 2: визначення ключа даних, що має використовуватись.

Індикатор фази ключа міститься у байті f14 і вказує, який ключ має бути використаний. Цей індикатор міститься в атрибуті CKA\_ID ключа і однозначно ідентифікує ключ. Обробка ключа, описана далі, має забезпечувати ефективний пошук ключа.

Операція 3: дешифрування включеного шифрованого повідомлення.

У CK\_MECHANISM використовується механізм CKM\_DES3\_CBC. Дешифруються 24 байти f15-f38 і перші 12 байтів дешифрованого результату виводяться поштовим ідентифікатором параметра. Після успішного дешифрування виконується операція 4, в іншому разі - операція 5.

Операція 4: формування і очищення хеш-значення.

Спеціальний 77-байтовий блок даних слугує основою формування хеш-значення дати.

Байти 1-53 відповідають першим 53 байтам матричного коду.

Байти 54-65 відповідають першим 12 байтам поточної нешифрованої частини повідомлення (поштовий ідентифікатор).

Байти 66-77 відповідають останнім 12 байтам поточної нешифрованої частини повідомлення.

Хеш-значення формується за допомогою SHA-1 і після цієї процедури перші 4 байти мають збігатись з байтами f54-f57. Якщо вони не збігаються, це означає, що дата не є дійсною. Якщо під час хешування виникає помилка або є незгідність у хеш-значенні, виконується операція 5.

Операція 5: повертання індикатора успіху.

Параметру bOk надається значення TRUE, якщо всі попередні операції були успішними, і значення FALSE, якщо при очищенні хеш-

значення були виявлені незгідності або якщо один з методів Pkcs 11 спричинив помилку. Повернуте значення має містити відповідне повідомлення про помилку або мати значення CKR\_OK у випадку відсутності помилок.

CK\_RV DE\_VerifyMsg (сеанс  
CK\_SESSION\_HANDLE)  
CK\_BYTE {} - int length  
CK\_BBOOL bOk  
Загальний блок даних має вигляд:

Блок даних для процедури верифікації				
П.п.	Довжина	№№ байт	Призначення	Опис
1	2	f1-f2	MsgType	Ідентифікатор ключового повідомлення, константа "КТ" або "КД"
2	1	F3	Version	Версія структури повідомлення, 1 байт, що починається з значення "01"
3	2	f4-f5	KeyGeneration	Лічильник генерувань прямим текстом, наприклад, "01"
4	4	f6	KeyID	Ідентифікатор ключа даних прямим текстом
5	4	f7-f11	KeyID	Ідентифікатор ТК прямим текстом
6	4	f12-f16	SigKeyID	Ідентифікатор ключа, що використовується для сигнатури
7	4	f17-f21	KTID	Ідентифікатор ТК, що використовується для шифрування проміжного повідомлення (див. п.2)
8	n-22	f22-f <sub>n</sub>	TranspKeyEncrypt HelperMsgEncrypt	Результат шифрування (згортання) ТК. Ключ даних: результат шифрування проміжного повідомлення
9	24	f <sub>n+1</sub> -f <sub>n+24</sub>	MAC	ТК: MAC для ключового повідомлення; формується як хеш-значення SHA-1 для полів 1+2+5+6+8 повідомлення, причому це значення шифрується сигнатурним ключем (механізм CKM_DES3_CBC_PAD, IV заповнюється нулями, ID - див. п.6). Ключ даних: MAC для ключового повідомлення; формується як хеш-значення SHA-1 для полів 1+2+4+3+6+7+8 повідомлення, причому це значення шифрується сигнатурним ключем (механізм CKM_DES3_CBC_PAD, IV заповнюється нулями, ID - див. п.6).

Невикористані поля заповнюються нулями. Режим виконання метода визначається параметром MsgType.

Генерований блок даних передається у повідомленні для MsgType ТК і ключа даних. Блок даних заповнюється з кожним прийнятим повідомленням.

Функція призначення CE\_VerifyMsg для MsgType ТК приймає повне транспортне ключове повідомлення в атрибуті MESSAGE, а його довжину у верхній частині шахти ліфта атрибуті LENGTH. Це вбудоване повідомлення слугує для забезпечення цілісності транспортного ключового повідомлення у реципієнта. Для активування верифікації виконуються такі операції:

Операція 1: формування вектора IV ініціалізації, який заповнюється нулями.

Операція 2: дешифрування прийнятого шифрованого повідомлення. У CK\_MECHANISM використовується механізм CKM\_DES3\_CBC. Дешифруються змінні поля MAC. При наявності помилок виконується операція 4.

Операція 3: очищення хеш-значення.

Хеш-значення дати формується з полів 1+2+5+6+8 транспортного ключового повідомлення і порівнюється з хешем операції 2. Хеш-значення формується з використанням SHA-1. Якщо хеш-значення є не ідентичними, то дата не є дійсною. Якщо під час хешування виявляється помилка або якщо виявлено незгідності у хеш-значенні, виконується операція 4.

Операція 4: повертання індикатора успіху.

Параметру bOk надається значення TRUE, якщо всі попередні операції були успішними, і значення FALSE, якщо при очищенні хеш-значення були виявлені незгідності або якщо один з методів Pkcs 11 спричинив помилку. Повернуте значення має містити відповідне повідомлення про помилку або мати значення CKR\_OK у випадку відсутності помилок.

Після виконання функції призначення CE\_VerifyMsg для MsgType ключа даних повне повідомлення ключа даних передається в атрибуті MESSAGE, а його довжина в атрибуті LENGTH. Це вбудоване повідомлення слугує для забезпечення цілісності транспортного ключового повідомлення у реципієнта. Для активування верифікації виконуються такі операції:

Операція 1: формування вектора IV ініціалізації, який заповнюється нулями.

Операція 2: дешифрування прийнятого шифрованого повідомлення. У CK\_MECHANISM використовується механізм CKM\_DES3\_CBC. Дешифруються змінні поля MAC. При наявності помилок виконується операція 4.

Операція 3: очищення хеш-значення.

Хеш-значення дати формується з полів 1+2+4+3+6+7+8 повідомлення ключа даних і порівнюється з хешем операції 2. Хеш-значення формується з використанням SHA-1. Якщо хеш-значення є не ідентичними, то дата не є дійсною. Якщо під час хешування виявляється помилка

або якщо виявлено незгідності у хеш-значенні, виконується операція 4.

Операція 4: повертання індикатора успіху.

Параметру bOk надається значення TRUE, якщо всі попередні операції були успішними, і значення FALSE, якщо при очищенні хеш-значення були виявлені незгідності або якщо один з методів Pkcs 11 спричинив помилку. Повернуте значення має містити відповідне повідомлення про помилку або мати значення CKR\_OK у випадку відсутності помилок.

Ці вбудовані методи обробки ключа мають включати імпортування згорнутого ключа і ефективний менеджмент. Мають бути імпортовані ключі типу CK, ключ даних, ТК.

```
CK_RV      CE_ImportKey      (сеанс
CK_SESSION_HANDLE,
довжина CKJLONG, CK_BYTE* HashValue
CE_EnumKeyKeyType
CK_CHAR [] KeyKID)
```

Функцію призначення CE\_ImportKey бажано виконувати, як це описано нижче.

Для згортання і розгортання використовується механізм CKM\_KEY\_WRAP\_WEBSENTRY. Отриманий ключ імпортується у криптообладнання розгортанням, причому ключ, який імпортується вдруге, тобто з тією ж фазою ключа, записується замість старого ключа.

Згорнутий ключ вноситься у параметр DATA, а його довжина у параметр LENGTH. Довжина ключа залежить від заповнення атрибутів ключа. Тип ключа верифікується параметром Key Type і обробляється відповідним чином.

Далі ключ вноситься у менеджмент ключа, що входить у кеш-операцію і дублікат-попередник, якщо він є, стирається.

Ключ даних дешифрується транспортним ключем ТК, ідентифікованим параметром KeyKID; для всіх інших типів ключа цей параметр не розглядається і заповнюється нулями.

Залежність від атрибута SKA\_END\_DATE є важливою. Він визначає кінцеву дату попередника ключа.

Атрибут імпортованого ключа містить рандомізоване число, через яке за допомогою SHA-1 формується хеш-значення. Це значення повертається у параметрі HashValue вбудованого метода і є необхідним для повідомлення підтвердження ключа.

У випадку появи помилки при застосуванні механізму розгортання повертається код помилки, в іншому випадку повертається CKR\_OK.

```
CK_RV      CE_GetKeyCount      (сеанс
CK_SESSION_HANDLE,
CE_EnumKeyKeyType,
int* counter)
```

Функція призначення CE\_GetKeyCount вказує, скільки ключів кожного типу зареєстровано у карті у менеджмент ключа, що входить у кеш-операцію. Завдяки цьому атрибути ключа можуть бути зчитані у сполученні з описаним нижче методом.

Цей метод визначається послідовністю:

```
CK_RV      CE_GetAttribute      (сеанс
CK_SESSION_HANDLE,
```

```
CE_EnumKeyKeyType,
CE_EnumKeyAttribute KeyAttribute,
int pos,
CK_BYTE [] * attribute,
int* length)
```

Тип ключа визначається параметром KeyType, і, отже, також таблицею, яку зчитують під час запису різних типів ключів у криптообладнання різними списками згідно з типом ключа.

Параметр KeyAttribute визначає атрибут, який має бути зчитаний, а параметр ITEM вказує початковий пункт у таблиці; спочатку з застосуванням метода CE\_GetKeyCount для всіх ключів або для одного типу ключа має бути отримане його максимальне значення. При виведенні атрибута SKA\_END\_DATE слід брати до уваги, що останній поточний ключ є теоретично нескінченно дійсним (до імпортування нового ключа того ж типу) і в атрибуті SKA\_END\_DATE містить кінцеву дату для попереднього ключа того ж типу, причому SKA\_END\_DATE вказує що вказаний ключ виведено.

Атрибут для дати має фіксовану довжину 8 байт, а атрибути CSK\_ID і SKA\_LABEL мають фіксовану довжину 128 байт. Якщо ці атрибути ключа для цих двох параметрів мають бути коротшими за 128 байт, решта байт заповнюється нулями. Отже, користувач завжди має достатньо пам'яті для реалізації цих методів. Якщо буфер користувача є занадто коротким, інформацію скорочують і повідомлення передається через CK\_RV.

```
CK_RV      CE_DeleteExpiredKey (сеанс
CK_SESSION_HANDLE,
CE_EnumKeyKeyType,
int* counter)
```

Функція призначення CE\_DeleteExpiredKey шукає карту для ключів з вичерпаним терміном дієвості і стирає їх. Ці ключі можна ідентифікувати тим, що їх системна дата є старішою за SKA\_END\_DATE наступного ключа (див.2.5.8); це також стосується ТК і СК. Можна проводити селективне стирання, використовуючи EnumType, а стерти карту повністю можна засобами SK\_KA (залишається лише ЛТК). Важливим є те, що цей спосіб не можна активувати під час імпортування ключа. Це бажано контролювати у вбудованому коді і вказувати у відповідному поверненому значенні. Завдяки цьому відвертаються будь-які побічні явища у менеджменті внутрішнього ключа.

Інтерфейс між системою забезпечення платежів і центром передачі вартостей (Поштовим Пунктом) бажано мати якомога вузьким, щоб виключити можливість маніпулювання через бічні канали.

Структура інтерфейсу між центром передачі вартостей (Поштовим Пунктом) і центральною системою забезпечення платежів ілюструється Фіг.3.

Центральна система забезпечення платежів надає інтерфейс для розподілення ключів, завдяки чому ключі, генеровані у компоненті KeyManagement центра передачі вартостей (Поштового Пункту), можуть розподілятися до криптосистем систем забезпечення платежів.

Центр передачі вартостей (Поштовий Пункт) надає системі забезпечення платежів інтерфейс вивільнення ключів, завдяки чому система забезпечення платежів може вивільняти ключ після його успішних розподілення і обробки.

Оскільки в обох проектах використовується, в основному, Java, рекомендовано реалізувати прикладний інтерфейс через Session Beans. Обслуговування для роз'єднання цих двох систем слід викликати засобами сервісу найменувань (JNDI).

Session Beans надає необхідну функціональність для імпортування даних ключа у центральну систему ZinS. Всі зв'язки показані у Фіг.4.

Фіг.4 ілюструє операції процесу інтегрування ключа даних у центральну систему забезпечення платежів (центральної ZinS).

Програма ImportKey переносить набір ключа даних у центральну ZinS.

Залежно від використання повідомлень ASN.1 ця програма обробляє прийняте повідомлення і викликає зберігання цього повідомлення у базі даних. Потім програма ImportKey ініціює розподілення даних ключа до локальних систем ZinS засобами CWMS.

Слід дотримуватись послідовності імпортування даних у базу даних і подальшого розподілення повідомлень. Цим гарантується, що дані будуть захищені ще до закінчення роботи з ними. Перевагою цього рішення є те, що воно полегшує відновлення інформації, яка постраждала від помилок, а у випадку втрати даних за необхідності забезпечується доступ до бази даних.

Параметри метода InsertKeyData ще не визначені, оскільки невідомо, чи підтримується формат ASN.1. Однак, цей метод може бути поповнений двома параметрами, які уможливають надсилання деталізованого повідомлення до Поштового Пункту.

У центральній ZinS цей спосіб розподілення забезпечує виконання таких функцій:

1. Архівування ключового повідомлення у центральному сервері ZmS

2. Розподілення ключових повідомлень від Поштового Пункту до індивідуальних поштових центрів через інтерфейс CWMS, наданий Vibris; структуру і використання сервісу CWMS описано в інструкціях до цього інтерфейсу.

3. Після завершення розподілення і імпортування в індивідуальні поштові центри генерується відповідне зворотне повідомлення.

Дані від центра передачі вартостей (Поштового Пункту) передаються у форматі ASN.1. Відповідне зворотне повідомлення також має бути у форматі ASN.1. Припустимими є також і інші формати. Адаптування до конкретного формату здійснюється належним синтаксичним аналізатором.

Бажаними форматами даних AN.1 є такі:

```
Rozpodiluvальне повідомлення для TK
TransportKeyMessage ::= SEQUENCE
{
  MessageType OCTET STRING, (fix 'KT')
  Version OCTET STRING, (0x01)
  KeyID OCTET STRING, (CKA_ID)
```

```
SigKeyID OCTET STRING, (CKA_ID
SignaturKey для MAC)
  TransKeyEncrypt OCTET STRING,
(TransportKey згортається ключем ЛТК)
  MAC OCTET STRING (MAC для всіх попередніх елементів)
```

```
}
Rozpodiluvальне повідомлення для TK
PostageKeyMessage ::= SEQUENCE
{
  MessageType OCTET STRING, (fix 'KD')
  Version OCTET STRING, (0x01)
  KeyID OCTET STRING, (CKA_ID)
  KeyGeneration OCTET STRING (0x01, зростання)
```

```
SigKeyID OCTET STRING, (CKA_ID
SignaturKey для MAC)
  TransKeyID OCTET STRING, (CKA_ID
TransportKey)
  DataKeyEncrypted OCTET STRING,
(PostageKey згортається ключем ЛГК і шифрується TransportKey)
  MAC OCTET STRING (MAC для всіх попередніх елементів)
```

```
}
(див. також 2.4.6)
Повідомлення вивільнення
KeyExchangeReceipt ::= SEQUENCE
{
  KeyID OCTET STRING, (CKA_ID прийнятого ключа)
```

```
  KeyLabelHash OCTET STRING, (хеш SHA-1
для CKA_LABEL прийнятого ключа)
  State BOOLEAN, (TRUE/FALSE активує/деактивує ключ)
  Message OCTET STRING (Опис успіху/неуспіху)
```

```
}
Можна встановити різні структури даних. Однак, структури, встановлені згідно з розглянутими втіленнями винаходу дають певні переваги, оскільки дозволяють скоротити передачу всієї суттєвої інформації. Зокрема, дані передаються через CWMS до локальних систем забезпечення платежів, які бажано розташовувати у поштових центрах провайдера поштових послуг.

```

При використанні формату ASN.1 дані спочатку компонують у внутрішнє ключове повідомлення, яке потім розподіляється, якщо повідомлення, визначені у цьому документі, були використані безпосередньо.

Пакети даних, прийняті через CWMS, відповідають ключовим повідомленням, визначеним раніше у цьому документі. Ці повідомлення потім піддають верифікації згідно з методом VerifyMsg вбудованого коду після адаптування до наведеної вище таблиці даних "Атрибути ключа даних". Після верифікації починається або імпортування ключа згідно з вбудованим кодом або генерування повідомлення про помилку (див. ключові повідомлення і стор.6-8 з порівнянням їх між собою для з'ясування структури даних, імпортованих методом CE\_ImportKey). Стирання старих ключів і оновлення виконуються автоматично згідно з описаними вище методами вбудованого коду.

Щоденно виконується метод `CE_DeleteExpiredKeys` для видалення з криптообладнання, якщо це необхідно, ключів з вичерпаним терміном дієвості. Під час імпортування цей метод гарантує стирання дублікатів ключів і їх заміну новими.

Методи вбудованого коду реалізуються з використанням криптоадаптерів (`CryptoAdapters`) класу Java, які надають всі функції (з тими ж найменуваннями), які надають вбудований код і інші методи PKCS 11.

За допомогою JNI використовується DLL (`CryptoAdapter.DLL`), який статично зв'язує DLL, надані від *Zaxus*. Таке статичне зв'язування додатково підвищує безпеку.

Застосування C++ для інтерфейсу JNI також забезпечує обробку помилок у кожному затребуваному сеансі (див. Стадію 3 "multithreading" у проектній документації PCFM. Робоча концепція підтримується тим, що криптообладнання ініціалізується у багатопотоковому режимі і після реєстрації у системі кожний учасник отримує власний сеанс (`getSession`, `C_GetSession`), завдяки чому учасник реєструється у DLL і обробка помилок встановлюється спеціально для нього. При реєстрації у системі спрямовується головний потік DLL і власна обробка помилок для виконання методів PKCS 11 і вбудованих методів.

Фіг.6 ілюструє огляд застосування. Бажано, щоб список ключів і код, що забезпечується вбудованими методами, видалялись. В іншому разі мають бути встановлені ідентичні структури.

Зв'язок DLL

Фіг 7 містить приклади бажаної інкапсуляції і зручної обробки помилок.

Застосування списків ключів є зайвим, якщо ці списки повністю лежать у вбудованому коді криптообладнання. Методи `GetAttribute` зводяться до простого ініціювання вбудованого метода `CE_GetAttribute`; відповідно, ініціювання імпортування і його застосування спрощуються, оскільки виконуються автоматично вбудованим кодом після перенесення необхідних даних.

Вбудований код містить розширений постійний список помилок.

Ключові повідомлення від Поштового Пункту зберігаються у базі даних центральної ZinS, щоб дозволити майбутню заміну дефектних локальних систем забезпечення платежів. Перш за все необхідно ввести повідомлення у базу даних і, по-друге, прийняти адміністративну інформацію.

Отже, у базі даних потрібно мати такі таблиці:

TransportKeyData	
SDItemNo	INT
Запис даних ключа	Див табл. на стор.14

Запис даних ключа містить ключове повідомлення у тому вигляді, в якому воно було прийняте Поштовим Пунктом; у випадку відмови локальної системи забезпечення платежів архівування ключового повідомлення дозволяє координувати дефектну локальну систему забезпечення платежів з іншими локальними системами забезпечення платежів шляхом імпортування архівованого ключа. Перед зберіганням повідомлення форматується згідно з ASN.1 для того, щоб передавати його до локальних систем.

Як модальність зберігання бажаною є спільна форма записів даних; це відповідає таблиці, яка описує блок даних, використаний у вбудованому методі `CE_VerifyMessage`.

TransportKeyManagement	
SNItem	Number
ReceiptDate	DATE (/ timestamp)
CompletionDate	DATE (/ timestamp)
DistributionDate	DATE (/ timestamp)
EndStatus	VARCHAR (20)
StatusText	VARCHAR (20)

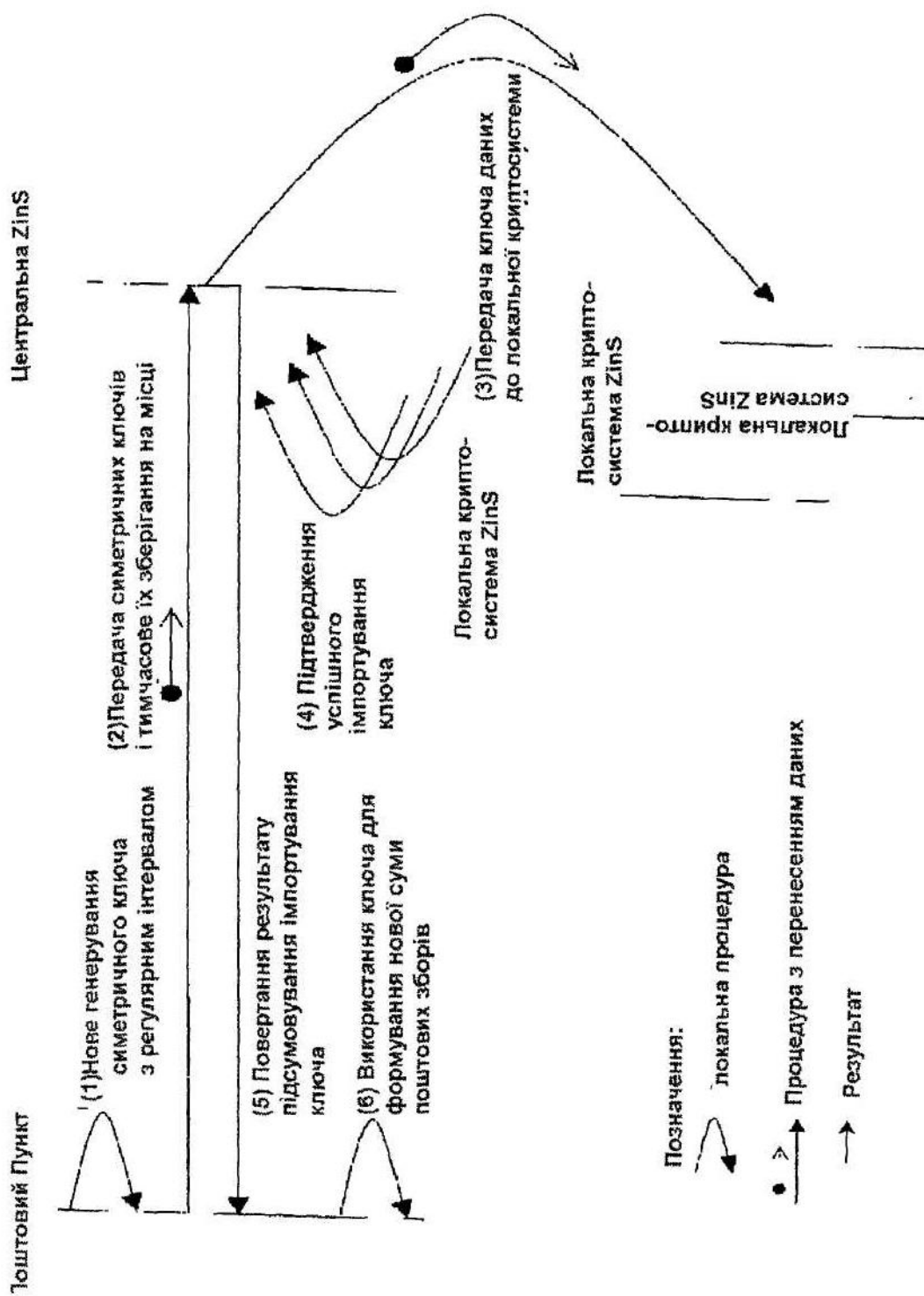
Цей запис даних має центральне значення. По-перше, він слугує для оцінювання зворотного зв'язку, прийнятого від локальної системи забезпечення платежів у поштових центрах, бажано, в усіх поштових центрах, інтегрованих у поштову систему провайдера поштових послуг, а також для проведення належного аналізу помилок і для формування тривожного сигналу для оператора системи у випадку перевернення часу під час процедури розподілення.

Відповідно, ця таблиця пов'язана з обробкою адміністративної інформації. Це уможливорює використання поля `SNItemNo` для призначення і оцінювання відповідної таблиці `TransportKeyData` і `TransportKeyReplayMessage` індивідуальних (83) поштових центрів, якщо це необхідно.

TransportKeyReplayMessage	
SNItem	Number
MailCenterNo	Number
MessageDate	DATE (/ timestamp)
MessageText	VARCHAR (520)

У цій таблиці індивідуальні ключові повідомлення-відповіді локальних систем забезпечення платежів архівуються у поштових центрах провайдера поштових послуг як функція процедур імпортування.

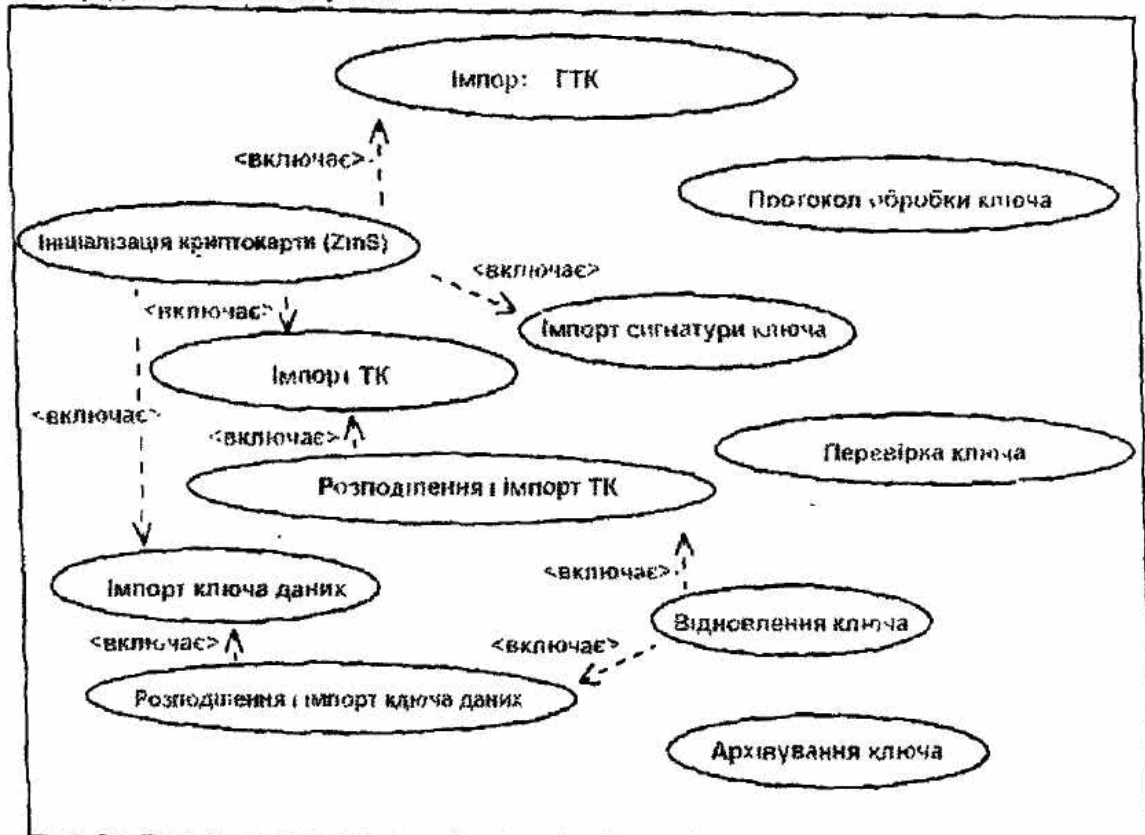
Фіг 1



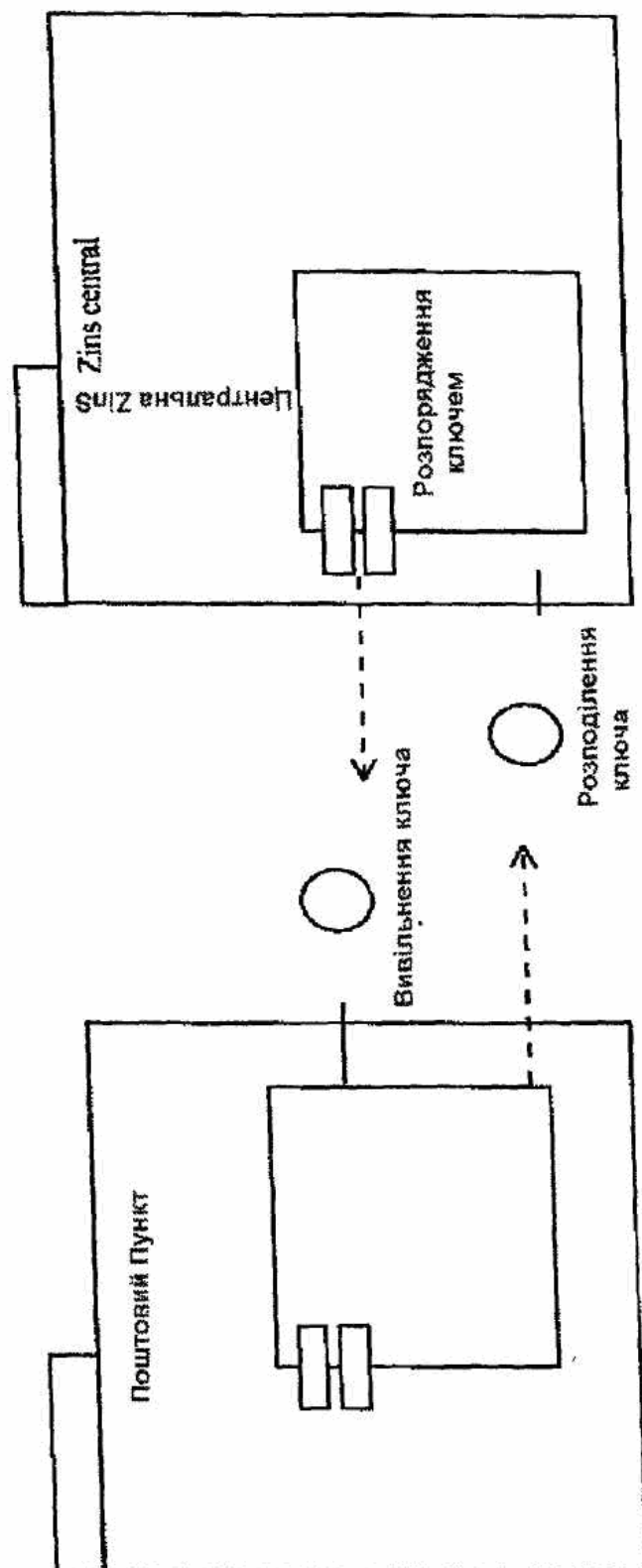
## Розпорядження ключами у Поштовому Пункті



## Розпорядження ключами у ZinS



Фіг 2



Фіг 3

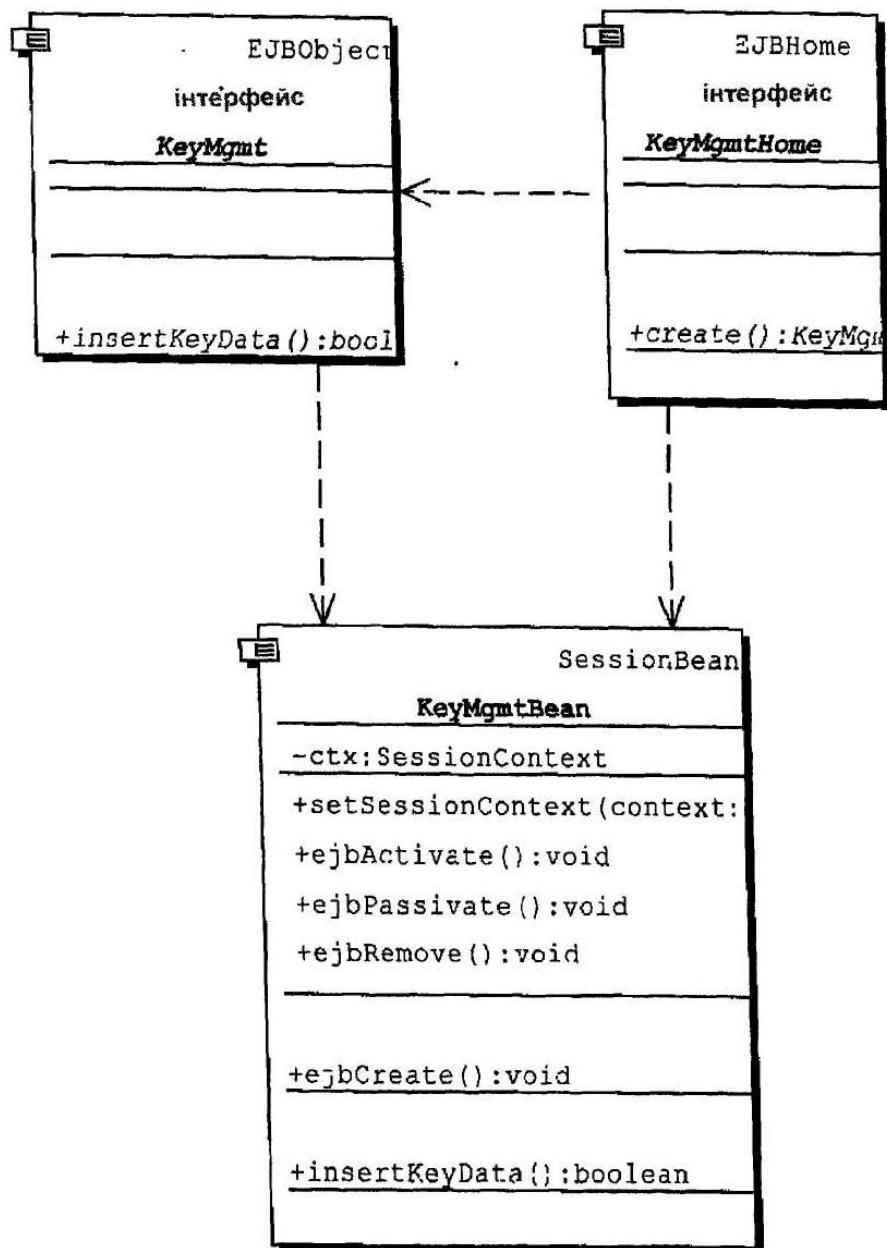
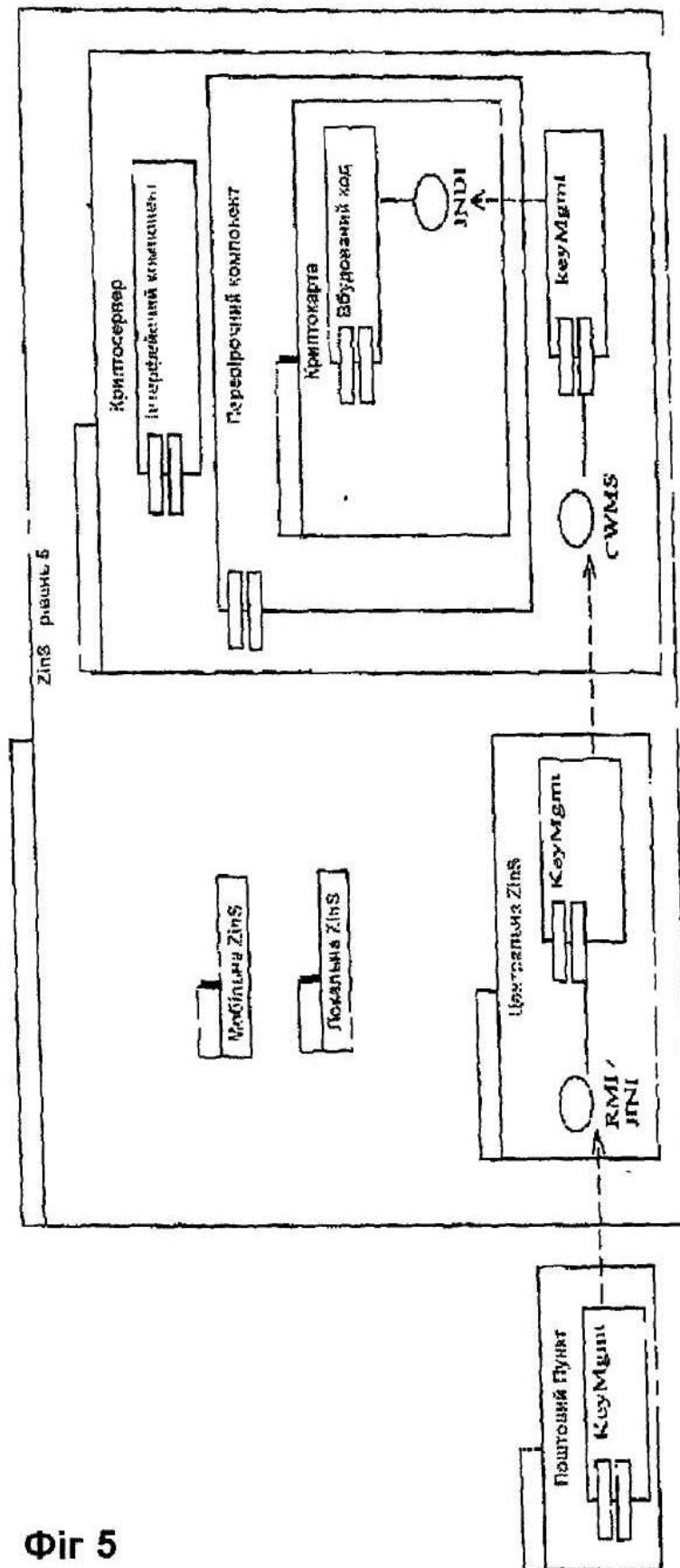
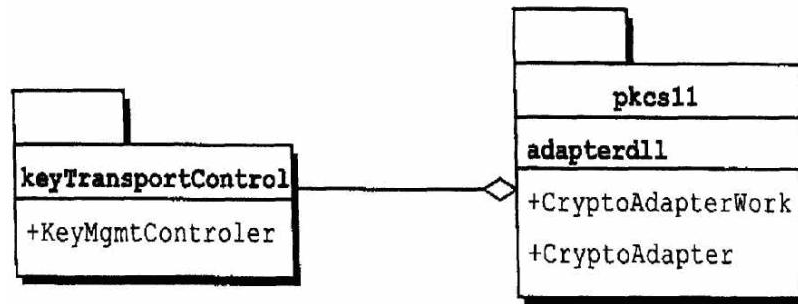


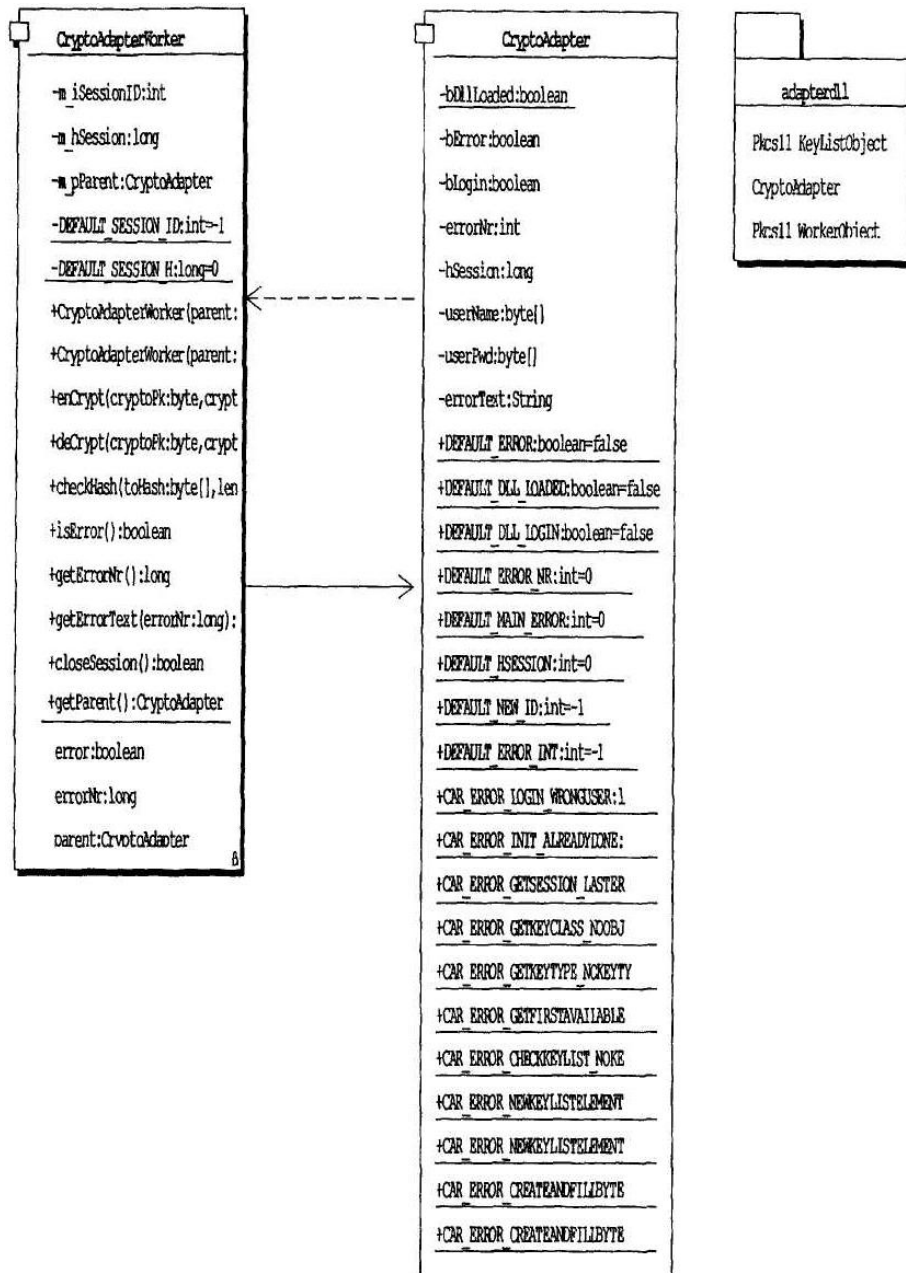
Fig 4



Фіг 5



Фиг. 6



Фиг. 7А

```

+CAR_ERROR_CREATESTRING_MOST
+DEFAULT_KEY_TEXT:String=new
+DEFAULT_ERROR_TEXT:String=n
-DEFAULT_USERNAME:byte[]=(ne
-DEFAULT_USERPWD:byte[]=(new
-IsErrorDll(id:int):boolean
-loginDll(hSession:long,userf
-logoutDll(hSession:long).box
-closeSessionDll(ID:int):boo
-getLogCountDll():int
-getSessionIDDll(hSession:lon
-getKeyLastCounterDll():int
-getErrorNrDll(id:int):long
-getASessionDll():long
-checkHashDll(id:int,toHash:byt
-enCryptDll(id:int,cryptoPk:byt
-deCryptDll(id:int,cryptoPk:byt
-getErrorTextDll(errorNr:long)
-getKeyClassDll(hSession:long)
-getKeyLabelDll(hSession:long)
-getKeyTypeDll(hSession:long)
-getKeyStartDateDll(hSession:lon
-getKeyEndDateDll(hSession:lon
-getKeyIDDll(pos:int):String
+CryptoAdapter()
+loadDll():boolean
+getLogCountNr():int
+IsDllLoaded():boolean
+login():boolean
+setUserName(newUserName:String)
+setUserPwd(newUserPwd:String)

```

Фіг. 7b

```

+logout():boolean
+getSession():CryptoAdapter
+isMainError():boolean
+getMainErrorNr():long
+getMainErrorText():String
+getKeyListCounter():int
+getKeyID(pos:int):String
+getKeyClass(pos:int):String
+getKeyLabel(pos:int):String
+getKeyType(pos:int):String
+getKeyStartDate(pos:int):String
+getKeyEndDate(pos:int):String
#enCrypt(id:int,cryptoPk:byte[])
#deCrypt(id:int,cryptoPk:byte[])
#checkHash(id:int,toHash:byte[])
#isError(id:int):boolean
#getErrorNr(id:int):long
#getErrorText(errorNr:long):String
#closeSession(ID:int):boolean

logCountNr:int
dllLoaded:boolean
userName:String
userPwd:String
ASession:CryptoAdapterWorker
mainError:boolean
mainErrorNr:long
mainErrorText:String
keyListCounter:int
keyID:String[]
keyClass:String[]
keyLabel:String[]
keyType:String[]
keyStartDate:String[]
keyEndDate:String[]

```

Фіг. 7c