

Занятие на сертификацию

за PKCS#10

- Запрос - это заголовок TBS-структуры (name, алгоритм подписи, signature)

Certificate Request Info := SEQ {

version INTEGER, // V1

subject Name, // бензин сертификата (имя)

subjectKeyInfo SEQ {

keyAlg AlgID,

key BIT STRING,

}

attributes [0] Attributes

* Рождение сертификата, который базируется на ключе

Challenge Password

* механическое закрытие носителя сертификата

или его генератора сертификата

или IX более информативно

использование языка программирования для

! Заміти за PKCS#10 є самонадісаними

Актор PKCS#10 вислов у режимі
нітійні заміти кого то, що все одно
замін є самонадісаним.

Підбідо у якості підп-я, що кер. Name
важливі ключі Key, заміні таємницються
Sk, що відповідає Key (pk).

Авторизація при заміні ключа виходить
за межі PKCS#10

Ключ видається сертифікат віртуалної CA
закінчено спроектувати не можна.

10-2 Замін на блокування /відхилення/
наповнення сертифікатів Гауптнадісаніх
така \Rightarrow з'явився СМР

Certificate Management Protocol (CMP)
протокол, який дозволяє керувати усім
життєвим циклом сертифікатів в мережі
насії інфраструктури.

RFC 4210, RFC 4211 (Certificate Request
Message Format, CRMF)

2005 рік

Протокол описує роботу 4х типів суб'єктів:

- 1) Кореневий центр $\xrightarrow{\text{root CA}}$ - може давати, центр,
актив підп-я всі інші сертифікати (самонадісн.)
- 2) інші УСЛ - не є головними, але
також можуть підп-тись
- 3) Центр реєстрації - може перевіряти
інших користувачів, але НЕ може видава-
ти сертифікати.
- 4) Підпісі суб'єкті (end entities)
 $\xrightarrow{\text{EE}}$
актив вони у даному перевірки
сертифікатів є кінцевими.

- Функціонал: Копійний функціонал обкладається кореневим центром новими типами північ-я (IX.26)
- 1) Ініціалізація Root CA - частково виходить за межі протоколу.
 - 2) Доведання нового ЦСК - включення нових сертифікатів, що мають новий-я ЦСК, вимагають дод. процедур
 - 3) Первинна ініціалізація ЕЕ - ІІРРОСПЕКЦІЯ Якщо користувача не було у базі, то він має пройти цієї можливості перевірки для отримання сертифікату
 - 4) Операції залог сертифікатами:
 - першість реєстрація та сертифікація клієнтів
 - створення ключа (+ сертифікат)
 - створення сертифікату без заміни ключа (напр., коли у нас змінилося ім'я)
 - створення ключа ЦСК (+сертифікату)
 - } створення ключа Кор-го центру - окрема ОКРЕМА СПЕЦІАЛІЗАЦІЯ \Leftarrow процедура
 - замініть на Крос-сертифікацію
 - створення Крос-сертифікатів
 - 5) Документування відомостей (discovery)

є окресленням функціональної для:

 - 1) розсилання сертифікатів, що відсилаються списків відповідальних CRL

Розсилання у двох режимах:

 - адресно
 - броадкаст (на усю мережу)
 - 6) Відновлення секретного ключа!

СПЕЦІАЛЬНО ДЛЯ СМР, АНЕ НЕОДНОДУХОВНО

ПРОБОТО функціонал, який при запуску якож відбіль. перевір-я, але дозв. буди. ключ.

7) Блокування, відкликання, погодження серт-ів

8) Підтвердження володіння sk! - специфічно для СМР

(Proof of Possession)

{ у PKCS#10 користувачі підб'єм sk та ін., що
підписували заміт, АЛЕ так можна єс-
буті лише з тими ключами, які пружа-
гли для підписів.

У СМР передбачені окремі процедури для:

- Ключів ЕЦП
- Ключів шифрування
- Ключів узгодження ключів (напр. Diffie-Hellman)

ЯК ЦЕ РОБЛЯТЬ?

B Утилізований спосіб

Усієв обмін за СМР є стандартизованим і
практично є обміном діялок за структуровани-

даних.

Форма СМР передбачає отримання іншого
механізму для захисту відповідей, що
передсилюються підтвердженнами.

Підтвердження може бути:

- * засланнями
- * шифр-зи
- * MAC-зображеннями

PKJ Messages := SEQ OF PKJMessage;

PKJMessage := SEQ {

header PKJHeader;

body PKJBody;

protection [0.. PKJProtection OPT]

extra Certs [1.. SEQ OF CMPCertificates]

Може protection містити заміт, підписану
то body є порожнім.

Дієм чи то protection може містити ког. обмінчи-
ти

P1ljHeader := SEQ {

PKI ^{база символів} ^{номер} **PVNO** INTEGER, // v2

Sender ^{базова ім'я} General Name,

receiver ^{загальне ім'я} General Name,

messageTime [0] GeneralizedTime,

protectionAlg [1] AlgID,

^{анулює повідомлення захисту, то після} **senderKID** [2] OCTET STRING,

receiverKID [3] OCTET STRING,

transactionKDD [4] OCTET STRING,

^{идентифікатор транзакції, який має викликати} **senderNonce** [5] OCTET STRING

^{анулює захист атаками на борем,} **receiverNonce** [6] OCTET STRING

freeText [7] SEQ OF UTF8 STRING

 небільші текстові побіг-я, що потрібно показати користувачу; це може не мати сенс, заживай тут пишуть що від користувача вимагається.

generalInfo [8] SEQ OF InfoTypeAndValue

^{идентифікатор} ^{інформація}

 може виключати що забов'єте, щоб тільки ідентифікатор був правильного формату, склад-я сподіванка інформація затином "ЦСК не може обробити засіт", скаже н-хб., і тоді

EoJ - [8]. OPTIONAL

P1ljBody:

- 2 б типів побіг-я
 - 1) Засіт та ~~sig~~ ^{TA} signature
 - 2) Засіт та ^{TA} signature та сертифікат

 Котре за протоколом CMP

- 3) ЗАМІСТ за PKCS#10
- 4) РОВ challenges/responses
чили "а ти відомий SK?"
- 5) Заміти та відповіді на додавання ключів
- 6) Заміти та відповіді блокування.
- 7) ЗІВ на відновлення ключів
- 8) ЗІВ на крос-сертифікацію
- 9) Пороменіння про стягнення ключів
(корен. центр, CA)
- 10) Пороменіння про публічну сертифікату,
пороменіння на відповіді, що його відмінили,
списки відмінених (CRL)
- 11) Дієвідповідність одержання сертифікату
- 12) Вкладене побудовлення
- 13) Побудовлення загального bugу і загальна
форма відповіді на нього
(General Message | General Response)
може бути замінено після функціонування.
- 14) Побудовлення про помилку
- 15) Дієвід - я то, що час сер - лу
- 16) Толочування: заміті / відповіді
на масове відсування

БІЛА ВОРОНА СМР

біє блокчейном ... АНЕ СТАМА АРТ 200 Spotify

Це відповіді на заміти обов'язково
починаються з поля STATUS, але у
СМР не є цим !

PKJStatusInfo := SEQ {

↑
status PKJ Status,

statusStr PKJFreeText OPT,

freeText freeText z Header

failInfo PKJFailureInfo OPT

ε 27 уточненных
поле

Итого якую у нас все хорошо, но
надеяется status PKJStatus = 0

и все, якую у нас не все хорошо,
то можна вказати чо саме у
PKJFailureInfo.

Синтаксис:

0 - все OK

1 - receiver не зміг виконати
занурення TAK, але яко
всією стоп-

зробив, але зміг зберегти з
текстом

Приклад поб-а статус 1:

sender відрівнув відповідьNonce, а
receiver не зміг це зробити, то
бін обробив замін, але не так
ак хотів користувач
(бінig 1).

2 - відмова (замін не виконано)

3 - очікування (приходьте після
а якщо я зміни)

4 - менш-я info відмінність ↪
(тесто якую бажає клієнте відмінність,
а бін не має правом. корист.), то бін
запишу: "NO-NO-NO!" ↪
також є

5 - надійність (одобрення) ↪
надійність

6) Конфідентність чи обмеження кіночі

(якщо користувач сам пам'ятає про заміну ключа, то він просить сервер не забувати проєкцію про обмеження).

Формат заміни на сертифікацію (CRMF):

Вони сконструювали нововідкритий шаблон, у якому можна задати будь-які поля майбутнього сертифікату і які зображення ми бажаємо використати.

! Поне "імені центру сертифікації" ми задаємо у шаблоні не можемо, а. поне "предмету", "німецькій дії", "сервісів ^{зберігання}"

- новий шаблон з усіма можливими полями сертифікату

Ми можемо сказати, що хочмо отримати сертифікат у ~~2022 р.~~

або зробити заміну ~~стародавніх~~

так само зробити і з завершеним строком дії і іншими ~~нарахованіми~~ ^{полями}

+ сервісна інформація:

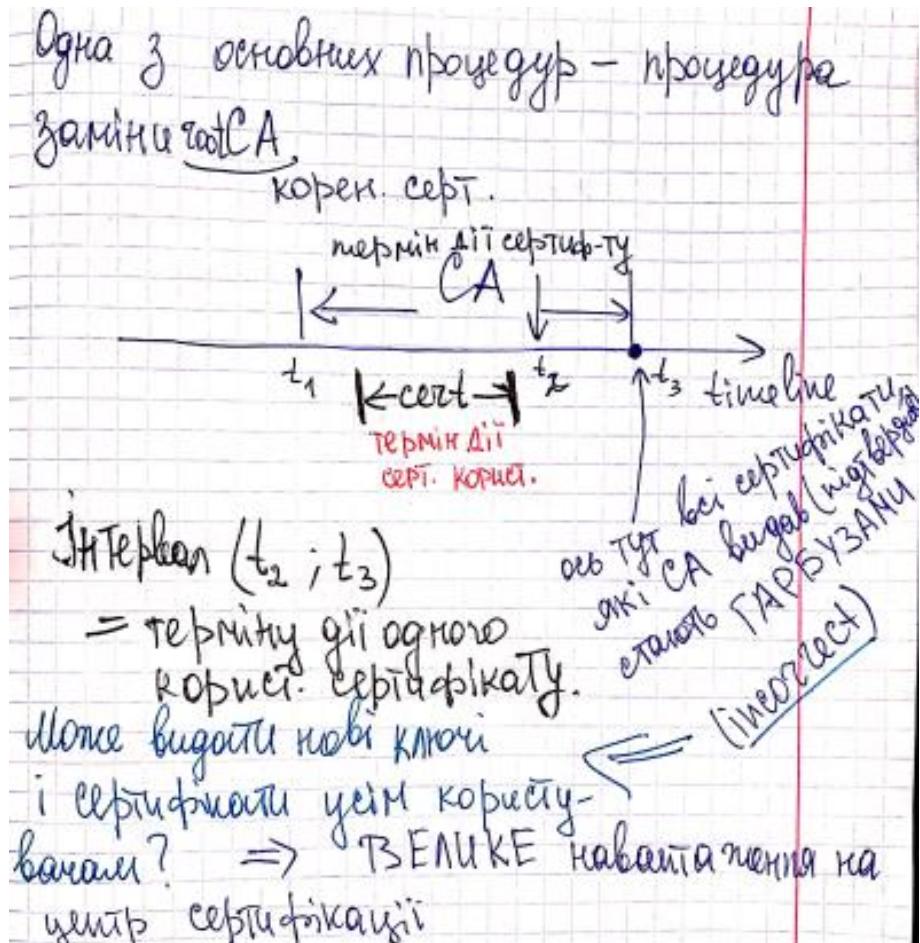
1) інф-я для автентифікації та реєстрації користувача

2) де ми бажаємо публікувати сертифікат та вказані чи хочмо ми його публікувати (LDAP, Web, ...) ^{по HTTP, IP}

3) попередній сертифікат - (у випадку користувача зареєстрованого в системі)

4) інформація про архівне зберігання

Процедура кореневого сертифікату:



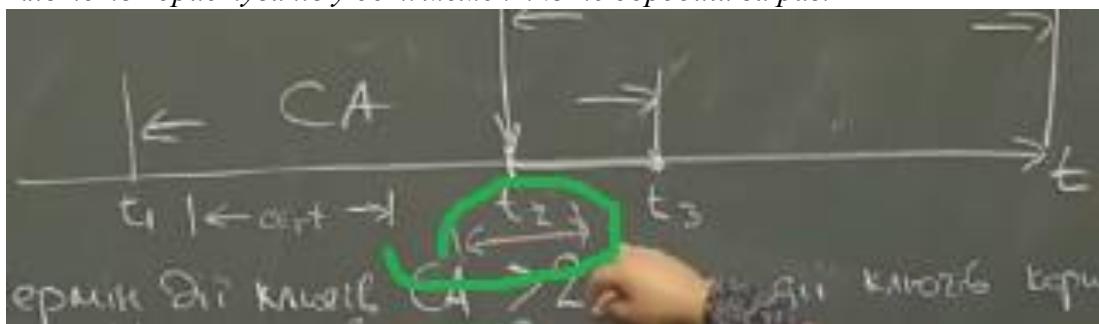
Тому постулюється таке:

- 1) Термін дії ключів центрів сертифікації (CA) повинен бути щонайменше у 2 рази довше за термін дії ключів звичайних користувачів.

В Україні звичайний користувач може мати сертифікат з терміном дії до 2х років, ЦСК до 5 років, а ЦЗО до 15 років. У кореневі міжнародні сертифікати видаються і на 30 років.

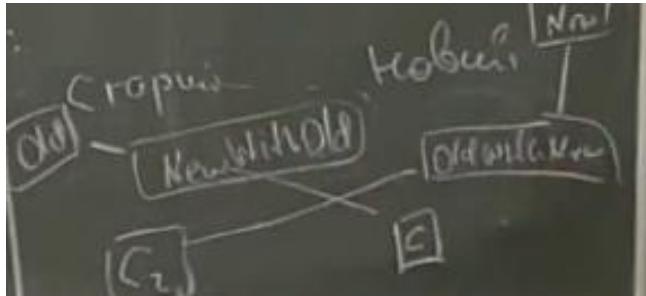
- 2) У момент t_2 CA видає новий ключ, а також два проміжніх сертифікати: *OldWithNew* (старий ключ, що підписан новим ключем), *NewWithOld*. Після цього старий ключ знищується, тобто його сертифікат залишається валідним протягом одного року, але ним більше не ставиться жодний підпис.

Далі для кожного користувача у якого тут (ДІВ. ЗЕЛЕНЕ) якийсь термін дії зійшов, CA видає новий сертифікат, що підписаний новим ключем. Так на проміжку ($t_2 ; t_3$) всі користувачі будуть замінювати собі сертифікати, адже у кожного з них завершується термін дії, а велику кількість користувачів у ост. момент t_3 не обробили зараз.



ОДНАК все одно будуть користувачі, які мають ще старі сертифікати із старими ключами, а також нові користувачі з новими. Для взаємодії таких користувачів нам потрібні *OldWithNew* та *NewWithOld*. Якщо у нас старий користувач, якому приходить новий сертифікат, то він підтверджується *NewWithOld* сертифікатом, який підтверджується старими ключами.

Тобто:



У момент t3 у всіх вже оновились сертифікати і нема потреби у *NewWithOld* та *NewWithOld*. САМОПІДПИСАНІЙ сертифікат НЕ збільшує довжину ланцюга! *NewWithOld* та *NewWithOld* самопідписані, тому *NewWithOld* та *Old* вважаємо єдиною сутністю.

У СМР розписано на 2 сторінки 8 випадків як оновлювати сковище кореневих сертифікатів, що робити у кожному випадку, а також як це робити.

У 2005р. в Україні – перший ЦЗО України, потім проблема з ключами у 2010 => у січні 2011 року встановлено термін дії ключа 15 років.

Користувач -> ЦСК -> ЦЗО

11-1 Синтаксис криптографічних повідомлень (Cryptographic Management System, CMS)

RFC 5652 – 5 редакція (фінал)

PKSC#7 – формат підписаного повідомлення, працює і сьогодні.

PKSC#8 – формат зберігання зашифрованих паролів, але замість паролів можна зберігати будь-які зашифровані дані.

З **PKSC#7** та **PKSC#8** відкрита інтернет-спільнота розробила першу версію **CMS 2315**, потім була **CMS 2630** і ін.

RFC 5652 є суттєво розширенім за **PKSC#7** : вона зберігає зворотну сумісність там, де це можливо, але це можливо не завжди.

RFC 5652 описує типи даних для криптографічних застосувань:

- *Data*
- *SignedData*
- *EnvelopedData* (запаковані дані, тобто тут мова йде про зашифровані контейнери, а саме зашифровані дані, що зберігаються з відомостями для розшифрування)
- *EncryptedData* (контейнер для шифрованих даних симетричними алгоритмами, жодних відомостей про ключ нема)
- *DigestedData* (геш від даних, вид спеціального контейнеру)
- *AuthenticatedData* (контейнер зі симетричним підписом та MAC-кодом, є простшим за *DigestedData*)

OKREMO EncrAndAuthData (спеціальне доповнення (не основний документ), яке і шифрує, і обчислює дані для автентифікації)

- 1) Усі дані у CMS мають стандартизовану форму:

Тип ContentInfo, що визначається ідентифікатором **1.2.840.113549.1.9.16.1.6. (СІІА, RSA)**

ContentInfo = SEQ{

contentType OID // що властно зберігається

content [0] ANY DEFINED BY contentType // парсером розбирається з тим, що записано, у залежності від того, що зберігається

}

ідентифікатор не стосується поля OID!

- 2) Data 1.2.840.113549.**1.7.1.** – pkcs#7, 1 – тип даних номер 1.

Якщо кодується ContentInfo таким ідентифікатором, то це означає, що у якості контенту йде все, що завгодно.

CMS нічого не вимагає жодної внутрішньої структури та навіть не вимагає щоби цей тип був якимсь DER-кодованим.