



УНИВЕРЗИТЕТ СИНГИДУМУ

**- МАСТЕР СТУДИЈСКИ ПРОГРАМ -
САВРЕМЕНЕ ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ**

**ЗАШТИТА ИНФОРМАЦИОНИХ СИСТЕМА
БАЗИРАНА НА
ИНФРАСТРУКТУРИ ЈАВНОГ КЉУЧА,
КРИПТОГРАФИЈИ И СМАРТ КАРТИЦАМА**

- мастер рад –

**Ментор:
Проф. др Младен Веинович**

**Студент:
Бранко Антић, дипл.инж.
Број индекса: 410386/2014**



Београд, 2015. године

Заштита информационих система базирана на инфраструктури јавног кључа, криптографији и смарт картицама

Сажетак: Предмет разматрања рада је заштита савремених информационих система коришћењем инфраструктуре јавног кључа и смарт картица, уз примену криптографских метода кроз криптографске алгоритме за генерисање кључева, технологије дигиталног потписа и дигиталних сертификата, са разматрањем њихових карактеристика, начина реализације и различитих примена. Реализована је апликација као сервис за потребе аутентификације корисника, валидације приступних права и потписивања фајлова помоћу смарт картице, на Java програмском језику и Java Smart Card платформи, уз осврт на основе поменутих технологија.

Кључне речи: безбедност ИС, инфраструктура јавног кључа, PKI, криптографија, криптографски алгоритам, криптографски кључ, дигитални потпис, дигитални сертификат, смарт картица, Java, Java Smart Card.

Protection of information systems based on public key infrastructure, cryptography and smart cards

Abstract: In this paper, the subject of discussion will be protection of modern information systems using the public key infrastructure and smart cards, with the use of cryptographic methods for cryptographic algorithms for generating cryptographic keys, technologies of digital signature and digital certificates, with consideration of their characteristics, ways of implementation and different applications. The implemented application as a service for the purposes of user authentication, validation, access rights and signing of files using smart cards, the Java programming language and Java Smart Card platform, in reference to the base of mentioned technologies.

Keywords: IS security, public key infrastructure, cryptography, PKI, cryptographic algorithm, cryptographic key, digital signature, digital certificate, smart card, java, java smart card.

Садржај

1.	Методологија научног истраживања.....	4 -
1.1.	Уводне напомене.....	4 -
1.2.	Предмет истраживања	4 -
1.3.	Циљ истраживања	4 -
1.4.	Истраживачке хипотезе	5 -
1.5.	Методе истраживања и ток истраживачког процеса	5 -
2.	Увод.....	6 -
2.1.	Облици угрожавања безбедности информационих система	6 -
2.2.	Безбедност и заштита информационих система	7 -
3.	Криптографски системи.....	8 -
3.1.	Симетрични криптографски алгоритми.....	8 -
3.1.1.	Криптографски мод	8 -
3.1.1.1.	Мод електронске кодне књиге.....	9 -
3.1.1.2.	Мод уланчавања блокова.....	9 -
3.1.1.3.	Мод повратног шифровања	10 -
3.1.1.4.	Излазни повратни мод	11 -
3.1.2.	Избор одговарајућег криптографског мода.....	11 -
3.1.3.	IDEA алгоритам	12 -
3.1.4.	AES стандард и Rijndael алгоритам	13 -
3.2.	Асиметрични криптографски алгоритми.....	13 -
3.2.1.	RSA алгоритам.....	14 -
3.3.	Дигитална енvelope	15 -
3.3.1.	Сесијски симетрични кључеви.....	15 -
3.4.	Криптографске компресионе функције.....	16 -
3.4.1.	MD5 алгоритам	17 -
3.4.2.	SHA1 алгоритам.....	17 -
4.	Технологије заштите информационих система	18 -
4.1.	Инфраструктура јавног кључа.....	18 -
4.2.	Сертификационо тело и дигитални сертификат	19 -
4.3.	Дигитални потпис	20 -
4.4.	Креирање и верификација дигиталног потписа.....	21 -
4.5.	Захтеви за сертификате и листе опозваних сертификата	22 -
4.6.	X.509 стандард дигиталних сертификата.....	23 -
4.7.	Стандарди криптографије јавног кључа	25 -
4.8.	Протоколи	26 -
4.8.1.	SSL протокол	26 -
4.8.2.	S/MIME протокол	27 -
5.	Смарт картице	28 -
5.1.	Историјат смарт картица	28 -
5.2.	Карактеристике и типови смарт картица	29 -
5.2.1.	Меморијске картице	30 -
5.2.2.	Микропроцесорске картице	31 -
5.2.3.	Контактне картице.....	31 -
5.2.4.	Бесконтактне смарт картице.....	32 -
5.3.	Хардверске карактеристике смарт картица	32 -
5.3.1.	Електронски контакт чипа смарт картице.....	32 -
5.3.2.	Микропроцесор смарт картице	33 -
5.3.3.	Копроцесори смарт картице.....	33 -
5.3.4.	Меморијски систем смарт картице	34 -

5.3.5.	Додатни хардвер	- 35 -
5.4.	Комуникација смарт картица са рачунаром.....	- 36 -
5.4.1.	Уређаји за комуникацију са смарт картицама.....	- 36 -
5.4.2.	Модел комуникације смарт картица и рачунара	- 36 -
5.4.2.1	APDU протокол.....	- 36 -
5.4.2.2	TPDU протокол.....	- 38 -
5.5.	Оперативни системи смарт картица.....	- 38 -
5.6.	Софтверско окружење	- 40 -
5.7.	Стандарди.....	- 40 -
6.	Java окружење	- 42 -
6.1.	Java платформа.....	- 42 -
6.2.	Java Cryptography Architecture	- 42 -
6.3.	Cryptographic Service Provider	- 43 -
6.4.	Security Policy.....	- 43 -
6.5.	Security Manager.....	- 43 -
6.6.	Java Secure Socet Extension	- 43 -
6.7.	Java Cryptography Extension.....	- 43 -
6.8.	Управљање кључевима и сертификатима.....	- 44 -
7.	Java Card технологија.....	- 46 -
7.1.	Java Card архитектура	- 46 -
7.2.	Java Card Runtime Environment.....	- 47 -
7.3.	Java Card Virtual Machine.....	- 47 -
7.4.	Формати фајлова	- 48 -
7.5.	Java Card Converter.....	- 48 -
7.6.	Java Card Interpreter.....	- 49 -
7.7.	Java Card Instaler и Off-Card инсталациони програм	- 49 -
7.8.	Системске класе	- 50 -
7.9.	Java Card Applets	- 50 -
7.10.	Пакети и конвенције именовања аплета	- 51 -
7.11.	Процес развоја аплета	- 51 -
8.	Реализација апликације.....	- 53 -
8.1.	Идеја решења	- 53 -
8.2.	Архитектура апликације	- 53 -
8.3.	Овлашћена лица	- 53 -
8.4.	Логички модел и имплементација.....	- 54 -
8.5.	Генерисање дигиталног сертификата и персонализација картице.....	- 57 -
8.6.	Аутентификација корисника	- 59 -
8.7.	Провера права приступа	- 61 -
8.8.	Читање података са картице.....	- 62 -
8.9.	Дигитално потписивање података.....	- 63 -
8.10.	Верификација дигитално потписаних података	- 63 -
9.	Закључак.....	- 64 -
10.	Списак слика, шема, дијаграма и табела.....	- 65 -
11.	Списак скраћеница.....	- 67 -
12.	Литература.....	- 69 -

1. Методологија научног истраживања

1.1. Уводне напомене

У овом истраживању обухваћене су технологије са циљем решавања различитих безбедносних проблема у функционисању савремених информационих система.

Тежиште је на инфраструктури јавног кључа, уз осврт на криптографске алгоритме, генерисање и дистрибуирање криптографских кључева, сертификациона тела, дигитални потпис и дигиталне сертификате.

Изложени су основни концепти електронских идентификационих докумената заснованих на технологији смарт картица.

Приказане су и основе *Java* и *Java Smart Card* платформи, за потребе реализације апликације за креирање дигиталног сертификата корисника и смештање на смарт картицу, као и коришћење смарт картице за потребе ауторизације, провере права приступа и електронског потписивања фајлова.

Дат је опис структуре и функционалности реализоване апликације и изнесени могући правци даљег развоја.

1.2. Предмет истраживања

Предмет овог истраживања су инфраструктура јавног кључа и смарт картице, њихова безбедност и примена за потребе генерисања криптографских кључева.

Обухваћени су појмови криптографије, методологије функционисања инфраструктуре јавног кључа са њиховим предностима и безбедносним проблемима, као и могућа решења.

Разматране су методологије генерисања криптографских кључева и дигиталних сертификата, са предлогом потенцијалних решења примене.

1.3. Циљ истраживања

Научни циљ: Истраживање, дефинисање и анализа функционисања инфраструктуре јавног кључа, њихових безбедносних проблема и потенцијалних решења, као и примене криптографских алгоритама за потребе генерисања криптографских кључева који се примењују у технологијама дигиталног потписа, дигиталног сертификата и смарт картица.

Друштвени циљеви: Информисање о примени инфраструктуре јавног кључа, његовим карактеристикама, безбедносним проблемима и решењима, начином употребе и валидације дигиталних сертификата при реализацији криптографске заштите тајности

и аутентичности, као и о могућностима примене у савременим информационим системима.

1.4. Истраживачке хипотезе

Општа хипотеза: Информациони системи су перманентно угрожени у области безбедности функционисања, поузданости трансакција и интегритета података. Као једно од адекватних решења, представљена је технологија инфраструктуре јавног кључа и смарт картица. Због својих карактеристика и широког дијапазона могућих примена, а самим тим и потенцијала у области заштите, представљају област истраживања од великог значаја.

Посебна/радна хипотеза: Инфраструктура јавног кључа и смарт картице, уз генерисање криптографских кључева и дигиталних сертификата представљају потенцијално решење за више безбедносних проблема.

1.5. Методе истраживања и ток истраживачког процеса

У раду су коришћене основне научне методе.

Метода дескрипције је коришћења при објашњавању основних појмова и система функционисања.

У теоријским сегментима су примењене методе индукције, дедукције и анализе, а у извођењу закључака методе синтезе и генерализације.

2. Увод

2.1. Облици угрожавања безбедности информационих система

Ера савременог живота и пословања делом је омогућена напредком информационих технологија, а самим тим и развој и примена све већег броја комплексних информационих система. Примена нових софтверских решења и информатичких технологија у пословним процесима, системима аутоматизације обраде података, као и процесима доношења одлука од стране менаџмента, постаје неизоставни део функционисања институција, предузећа и појединаца [1].

С обзиром на функције система да врше прикупљање, чување, обраду и анализу података, уређују и контролишу процесе аутоматизације и електронске трансакције, то су они перманентно угрожени са аспекта заштите безбедности података и функционисања. Безбедност података је од изузетног значаја за сваки пословни систем. Информације, интелектуална својина, документација, садржај, мреже, подаци о запосленима и корисницима итд, могу се сматрати делом кључних добара корпорације. Сваки од ових елемената мора да буде брижљиво чуван у циљу обезбеђења поверљивости пословних тајни једне организације.

Постоји више врста угрожавања безбедности у информационим системима [2]. Према циљу нападача, претње је могуће класификовати на следећи начин:

- **модификација или уништавање података,**
- **нарушавање приватности** - откривање информационог садржаја и
- **неауторизовано коришћење рачунарских ресурса.**

Модификација или уништавање података је опасност која се односи на могућност приступа интерним меморијским ресурсима рачунара са безбедносно критичним информационим садржајима.

Нарушавање приватности је претња која се односи на неауторизовани приступ поверљивим информацијама. Један од напада ове врсте је прислушкивање (*eavesdropping*), које је веома тешко на време открити јер нема видљивих спољних манифестација. Прислушкивање је веома актуелна врста напада, јер се велика количина информација у отвореном облику размењује путем потенцијално небезбедних канала. Лажно представљање (*masquerading*) је врста напада у којој нападач покушава да искористи туђи идентитет ради напада на рачунарски систем.

Неауторизовано коришћење рачунарских ресурса се најчешће испољава кроз укидање сервиса (*denial of service*). Основни циљ ове врсте напада је нарушавање исправног рада расположивих сервисних програма у систему. Напади наведеног типа користе злонамерне програме за генерисање великог броја захтева за успоставу конекције, који ће бити одбијени. Овим долази до преоптерећења мрежног сервера који, након успешно покренутог напада, или прекида рад или само одговара на захтеве из напада и тиме је онемогућено да му легални корисници приступе.

Посебна претња информационим системима подразумева класу напада који се изводе без нарочитог циља од стране нападача, тј. без неке експлицитне користи.

Криптоанализа је напад којим се остварује претња нарушавања приватности. Активност нападача је усмерена откривању тајних параметара заштићених комуникација. Могућност криптоанализе је у највећој мери одређена начином чувања и размене симетричних и асиметричних кључева, дужином кључева, снагом примењеног криптографског система и рачунарским ресурсима нападача.

Напад типа **тројанског коња** представља дистрибуцију злонамерних програма на радне станице у рачунарској мрежи. Ова врста напада се обично темељи на слабостима самог оперативног система. Тројански коњ се кориснику приказује као легалан програм, али је пројектован тако да изврши додатну нелегалну активност. Ова врста рачунарског вируса се обично маскира у системске програме и брише трагове сопствене инсталације. Једини траг остаје у измењеној величини (и функционалности) одговарајућих системских програма.

2.2. Безбедност и заштита информационих система

У системима са традиционалном организацијом, мере безбедности се базирају на ограничавању неауторизованог приступа информацијама и заштићеним деловима система. Међутим, овај модел није погодан за дистрибуирано мрежно окружење, имајући у виду његову основну карактеристику да омогућава отворену и што доступнију комуникацију [3]. Зато су, паралелно са развојем рачунарских система, развијани и софтверско - хардверски системи заштите чија је улога да онеспособе, отклоне или умање претње и потенцијалне безбедносне пропусте.

Неки од начина одбране су:

- шифровање, за заштиту тајности података,
- вишестепена антивирусна заштита,
- правилна употреба мрежних баријера, које врше филтрирање *IP* пакета,
- примена технологије дигиталног потписа, која омогућује заштиту интегритета поруке, проверу идентитета пошиљаоца и обезбеђује непорецивост садржаја послате поруке,
- процедура узајамне аутентификације субјеката комуникације,
- коришћење дигиталних сертификата, као параметара који јединствено гарантују идентитет субјекта и
- коришћење одговарајућег криптографског хардвера за чување и генерисање кључева и извршавање безбедносно критичних процедура.

Широк је дијапазон система за заштиту информационих система, и у већини случајева се користе криптографски механизми заштите. Савремена решења система заштите карактерише вишеслојна архитектура, са употребом хардверских безбедносних модула (*HSM - Hardware Security Module*) као што су смарт картице и криптографски копроцесори, а базирају се на инфраструктури јавног кључа и примени технологија криптографије, дигиталног потписа и дигиталних сертификата [4].

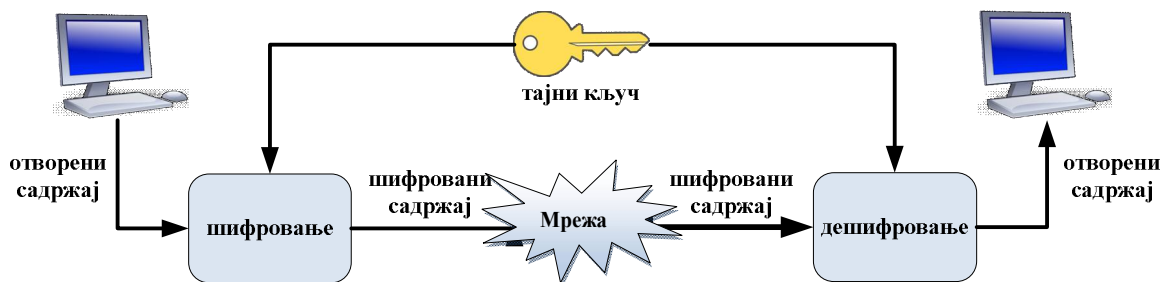
3. Криптографски системи

3.1. Симетрични криптографски алгоритми

Симетрични криптографски алгоритми су у криптографији почели први да се користе. У почетку су имали улогу у пословима државних структура, па су из тог разлога били тајни, креирани и коришћени управо од стране одговарајућих државних органа. Са развојем информационих технологија и комуникације, јавила се потреба за јавним криптографским алгоритмима. Данас се углавном користе у дистрибуираним системима електронских трансакција [5].

Код симетричних криптографских алгоритама се користи исти тајни кључ за шифровање и дешифровање, па се ова врста алгоритама другачије и назива - алгоритми са тајним кључем.

При употреби симетричних криптографских алгоритама, као параметар у комуникацији бира се одговарајући алгоритам који ће се користити током трансакције. Алгоритми се најчешће примењују у облику листе имплементираних криптографских алгоритама.



Шема 3-1 Симетрични криптографски алгоритми

Постоје две врсте симетричних криптографских алгоритама [6]:

- блок шифарски криптографски системи (*block cipher, bulk cipher*) и
- секвенцијални шифарски системи (*stream cipher*).

Блок шифарски системи врше криптовање отвореног текста (*plain text*) и шифроване поруке (*cipher text*) у блоковима одређене величине.

Секвенцијални шифарски системи врше криптовање низа бита, бајтова или речи.

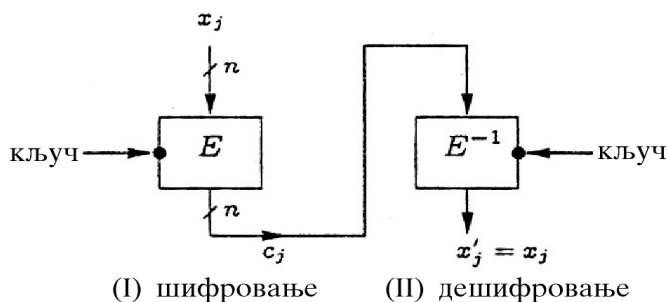
3.1.1. Криптографски мод

Дефинисан је велики број јавних симетричних алгоритама који се користе као стандарди у савременим системима заштите. Током примене уочене су извесне опште слабости карактеристичне за употребу блок шифарских система у специфичним условима. Ове слабости се отклањају комбиновањем неколико криптографских модела.

Криптографски мод представља начин употребе базичног шифарског алгоритма и најчешће је комбинација повратне петље и додатне једноставне операције. Операције које се користе у различитим криптографским модовима су једноставне, јер је поузданост система условљена поузданошћу основног шифарског алгоритма, а не примењеним криптографским модом [7].

3.1.1.1 Мод електронске кодне књиге

Мод електронске кодне књиге (*ECB - Electronic Codebook Mode*) представља најједноставнији и најбржи начин употребе симетричних криптографских алгоритама. Применом *ECB* мода, сваки блок отвореног текста се трансформише у блок шифрата, без постојања било какве повратне спреге [8].



Шема 3-2 Мод електронске кодне књиге

Опасност у употреби овог мода се налази у чињеници да уколико криптоаналитичар има отворени текст и шифрат неколико порука, може направити електронску кодну књигу, без познавања кључа. Исте отворене поруке ће имати исти шифрат, па се простом заменом блокова шифрата добија отворени текст. Понављање одређених фрагмената порука је реална појава, а слабости *ECB* мода нема уколико се увек шифрују различите поруке.

Проблем ове природе се елиминише на неколико начина:

- применом компресионих метода, без губитка информационог садржаја (различите компресионе методе се управо заснивају на препознавању битских сегмената који се понављају и на елиминисању њиховог понављања) и
- применом других криптографских модова.

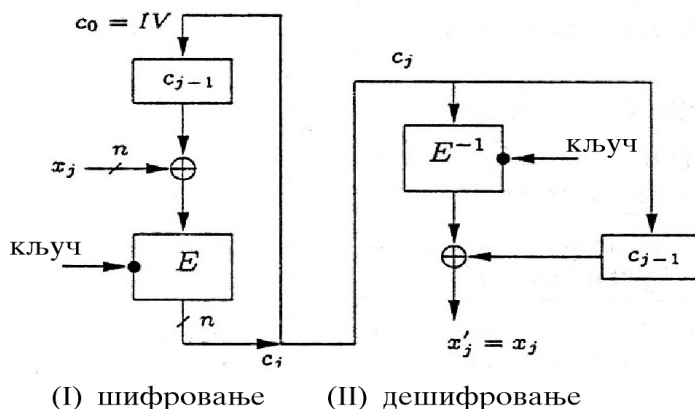
3.1.1.2 Мод уланчавања блокова

Мод уланчавања блокова (*CBC - Cipher Block Chaining Mode*) повезује блокове шифрата тако што се резултат шифровања претходних блокова користи при шифарској трансформацији текућег блока. Наведеним поступком се постиже да сваки блок шифрата не зависи само од текућег блока, већ и од свих претходно шифрованих блокова отвореног текста [8].

У *CBC*, блокови се међусобно повезују тако што се извршава операција „ексклузивно или“ (*XOR - Exclusive Or*) између текућег блока отвореног текста и претходно шифрованог блока, а затим се тако добијени блок шифрује. Процес дешифровања се одвија на инверзан начин: добијени блок података се дешифрује, а

затим се примени операција XOR над тако добијеним блоком података и претходним блоком шифрата.

Карактеристика CBC мода је да се идентични блокови отвореног текста шифрују у различите блокове шифрата. У првом кораку шифровања и дешифровања неопходно је коришћење иницијалног вектора, чија је величина једнака величини блока који се шифрује, односно дешифрује. Иницијални вектор не мора бити тајни податак, иако је погодно да се генерише на случајан начин. Овим се додатно повећава безбедност система.

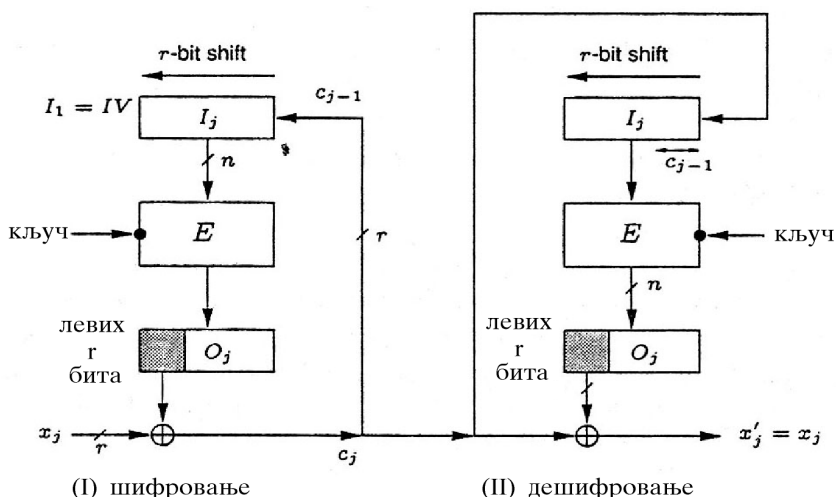


Шема 3-3 Мод уланчавања блокова

Постоји модификација овог мода, тзв. мод уланчавања блокова са пропагацијом ($PCBC$ - *Propagating Cipher Block Chaining*), где се суштина огледа у примени ексклузивне дисјункције на текући блок отвореног текста, претходни блок шифрата, али и на претходни блок отвореног текста.

3.1.1.3 Мод повратног шифровања

У одређеним применама се јавља потреба да се делови отвореног текста шифрују и преносе у јединицама величине мање од основне величине блока, што се реализује **модом повратног шифровања** (CFB - *Cipher Feedback Mode*). Овај мод се означава као r -bit CFB мод, где је r мање или једнако од величине блока примењеног блок шифарског алгорита [8].



Шема 3-4 Мод повратног шифровања

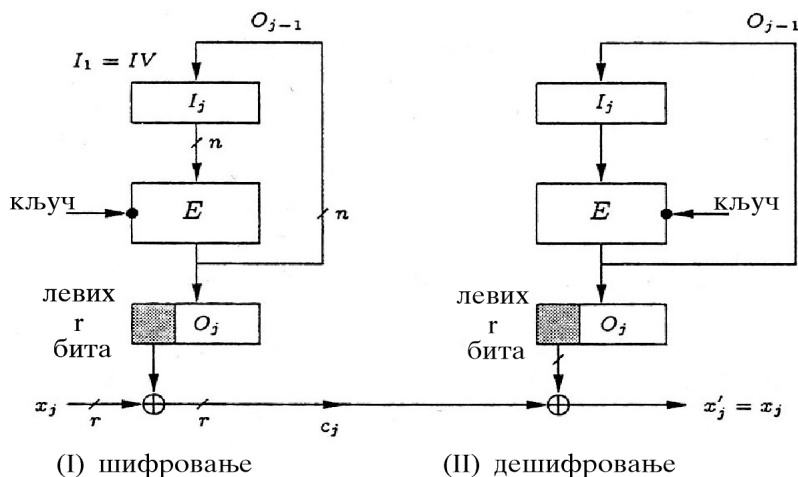
Операција шифровања се изводи на следећи начин:

- отворени текст се подели на блокове величине $r\text{-bit}$, формира се иницијализациони вектор величине $n\text{-bit}$ и упише у повратни регистар и одабере се кључ шифарске трансформације,
- формира се излазни блок тако што се применом одабраног кључа изврши шифровање садржаја повратног регистра,
- блок шифрата се формира тако што се примени операција ексклузивне дисјункције над садржајем текућег блока отвореног текста и $r\text{-bit}$ најмање тежине излазног блока,
- садржај повратног регистра се помера за r бита улево, а на место r бита најмање тежине се уписује формиран блок шифрата.

Дешифровање се одвија на сличан начин. Једна од карактеристика *CFB* мода је да се и на пријемној и на предајној страни основни шифарски алгоритам користи у моду за шифровање.

3.1.1.4 Излазни повратни мод

Излазни повратни мод (*OFB - Output Feedback Mode*) представља спој добрих особина *ECB* и *CFB* мода, који спречава пропагацију грешака и омогућава пренос података у јединицама мањим од основне величине блока. Уколико се деси грешка на једном биту приликом шифровања отвореног текста, применом излазног повратног мода, неће бити пропагације грешке, већ ће примена шифарског алгоритма за дешифровање резултирати у једном погрешном биту дешифрованог отвореног текста



Шема 3-5 Излазни повратни мод

Шифарска трансформација отвореног текста се одвија на сличан начин као у *CFB* моду. Разлика је у томе што се повратна спрега остварује коришћењем садржаја излазног блока, док се код *CFB* мода остварује коришћењем блока шифрата [8].

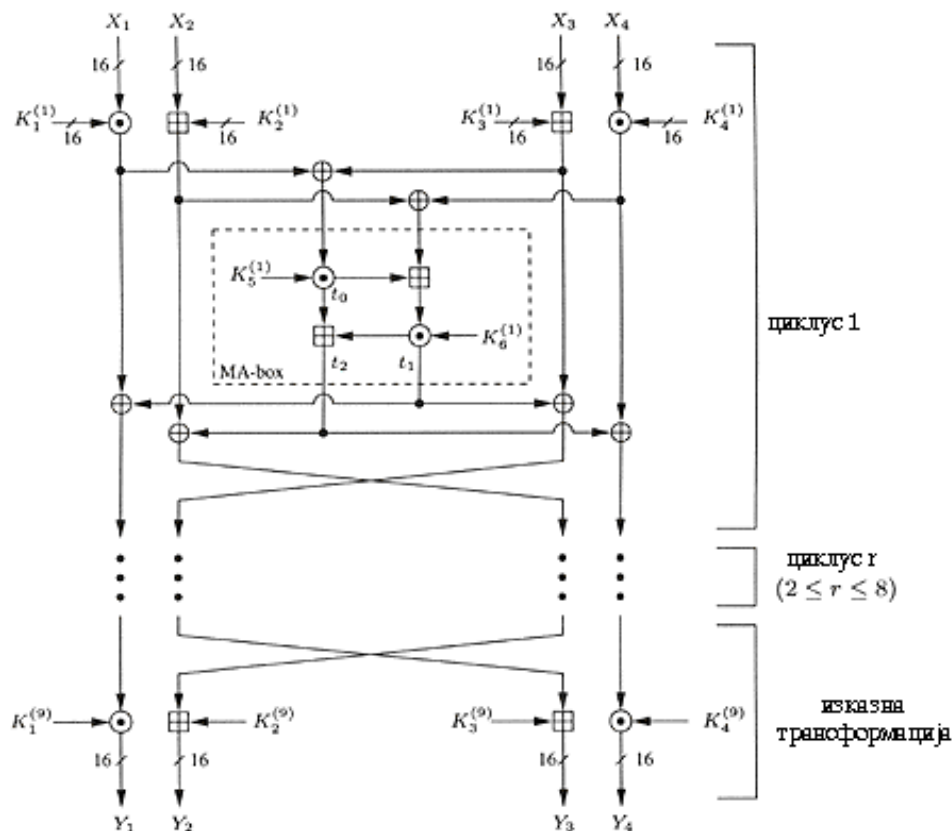
3.1.2. Избор одговарајућег криптографског мода

Избор конкретног мода примене симетричног криптографског алгоритма у највећој мери зависи од захтева корисника. Уобичајено је да се за апликације где је

императив брзина користи *ECB* мод. Овај мод се безбедно користи и приликом размене кратких порука, јер при том не долазе до изражаја његове слабости. *CBC* мод се најчешће користи за шифровање датотека. Мод *CFB* се често користи за шифровање низова карактера, у којима се сваки карактер индивидуално третира. *OFB* криптографски мод се често користи у системима великих брзина, где се не дозвољава пропагација грешака [8].

3.1.3. IDEA алгоритам

Један од најзаступљенијих савремених симетричних криптографских алгоритама у комерцијалним апликацијама је **IDEA алгоритам** (*International Data Encryption Algorithm*). Блок података који се шифрује је дужине 64-bit, док је кључ примењен у трансформацији дужине 128-bit. Доња граница сигурности за примену симетричних криптографских алгоритама у савременим условима је управо 128-bit. Трансформација отвореног текста се одвија кроз осам циклуса и по завршетку последњег циклуса добија се шифровани текст [9].



Шема 3-6 IDEA алгоритам

IDEA криптографски алгоритам је оптимизован за извршавање на 16-bit процесорима. Све елементарне операције се изводе над 16-bit подацима. Процес дешифровања се врши на исти начин као процес шифровања, али припрема кључа за дешифровање знатно дуже траје.

IDEA алгоритам се заснива на инвентивном спрезању алгебарских операција у различитим доменима, погодним за софтверску имплементацију.

3.1.4. AES стандард и Rijndael алгоритам

У току 2001. године, амерички национални институт *NIST* (*National Institute of Standards and Technology*) је за симетричне криптографске алгоритме објавио **AES стандард** (*Advanced Encryption Standard*), који је требао да замени претходни стандард *DES* (*Data Encryption Standard*), из разлога јер су се поруке шифроване према *DES* стандарду могле лако дешифровати, због недостатака у самом алгоритму, мале дужине кључа и повећане процесне моћи рачунара [8].

Након селекционе процедуре, за реализацију *AES* стандарда изабран је *Rijndael* алгоритам који су реализовали белгијски математичари *Joan Daemen* и *Vincent Rijmen*.

Rijndael је блок шифарски алгоритам који подржава трансформацију блока отвореног текста променљиве дужине (128/192/256-bit), применом кључа променљиве дужине (128/192/256-bit). За реализацију *AES* стандардног алгоритма изабрана је варијанта *Rijndael* алгоритма која подржава трансформацију блока података од 128-bit, са променљивом дужином кључа. Алгоритам *AES* реализује операција шифровања и дешифровања блока података у променљивом броју циклуса. Број циклуса зависи од дужине кључа и износи 10/12/14 за величину кључа од 128/192/256-bit, респективно.

У оквиру једног циклуса, трансформација се састоји од четири корака:

- *ByteSub* - нелинеарна супституција бајтова,
- *ShiftRow* - циклични померај врста матрице,
- *MixColumn* - множење колона матрице фиксним полиномом по модулу и
- *RoundKey* - сабирање кључа рунде са матрицом.

3.2. Асиметрични криптографски алгоритми

Асиметрични криптографски алгоритми представљају савремени вид примене криптографије. Криптографија асиметричног кључа користи различите кључеве за енкрипцију и декрипцију, па су алгоритми из те групе познати и као криптографски алгоритми са јавним кључем [5].

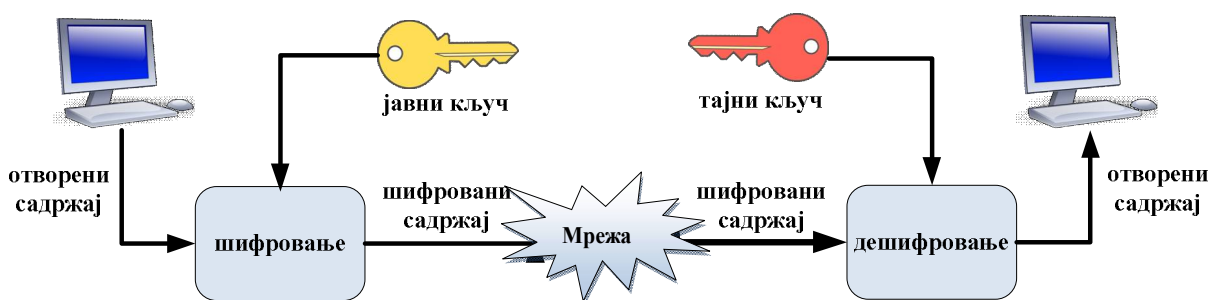
Потребно је истаћи значајне карактеристике асиметричних алгоритама:

- уколико се изврши шифровање поруке применом јавног кључа одређеног корисника, њено дешифровање може се једино извршити применом приватног кључа,
- уколико се изврши шифровање поруке применом приватног кључа одређеног корисника, њено дешифровање може се извршити једино применом одговарајућег јавног кључа и
- на основу познавања тајног кључа могуће је одредити јавни, док је на основу познатог јавног кључа немогуће одредити тајни кључ.

Ова врста криптографских алгоритама је произашла из потребе дистрибуције криптографских кључева, који је био главни безбедносни проблем приликом употребе симетричних криптографских алгоритама, па су првобитно развијени за заштиту тајности.

Безбедност ових алгоритама заснива се на рачунској сложености одређених математичких операција које се примењују у датим системима. Међутим, висока рачунарска захтевност ових поступака утиче на перформансе система у којима се користе, тако да се не препоручује примена за заштиту тајности информација у системима са великим протоком информација.

У савременој криптографији асиметрични алгоритми често се користе за реализацију размене симетричних кључева у апликацијама које примењују симетричне криптографске алгоритме (нпр. размена сесијских кључева). Применом асиметричних криптографских алгоритама могуће је остварити функције провере интегритета, аутентичности и непорецивости. Такође, најчешће се примењују у технологији дигиталног потписа и дигиталне енvelope.



Шема 3-7 Асиметрични криптографски алгоритам

Са становишта квалитета, отпорности на различите врсте напада и начина имплементације, алгоритми са јавним кључем нису подједнако ефикасни.

Најчешће коришћена алгоритма из ове групе су:

- **RSA** (*Rivest, Shamir, Adleman Algorithm*) и
- **Diffie-Hellman** - Познат је као *key-agreement algorithm* с обзиром да се не може користити за енкрипцију јавног кључа. Алгоритам се користи како би омогућио двома странама да генеришу тајни кључ делећи информације кроз јавни медиј. Тај кључ се онда може користити за енкрипцију алгоритмом тајног кључа.

3.2.1. RSA алгоритам

RSA алгоритам је најпознатији представник групе асиметричних криптографских алгоритама. Настао је осамдесетих година прошлог века, а назив је добио по иницијалима својих аутора: *Rivest, Shamir, Adleman* [10].

RSA алгоритам се заснива на проблему факторизације великих бројева на просте чиниоце. Први корак у реализацији алгоритма је избор бројева p и q . Према жељеном степену сигурности и намени система, одређује се величина бројева p и q (најчешће изражена у битима). Потом се на случајан начин бира број p_1 за који се врши провера да ли је прост. Уколико се утврди да је број прост тада је $p = p_1$, у противном се бира нови број p_1 . Поступак се рекурзивно понавља. Када се реализује селекција броја p на исти начин се одређује вредност q . Након одабира наведених величина врши се избор јавног и тајног кључа, e и d .

Проблем генерисања случајних бројева у својој основи је проблем генерисања случајних бита, при чему се издвајају два основна приступа [11].

Први начин се базира на **генераторима псеудослучајних низова** (*CSPRNG – Cryptographically Secure Pseudo-Random Number Generator*) имплементираних у рачунарским системима који као иницијални извор случајности користе одређена стања у систему [12]. Најчешће су то информације везане за време између два откуцаја на тастатури, садржај временског регистра, текуће позиције показивачких уређаја, садржаји променљивих табела унутар оперативног система итд. Карактеристика наведеног начина генерисања псеудослучајних низова је да његова битска неодређеност није већа од битске величине иницијалних података. Уколико иницијални подаци нису у довољној мери случајни, као што је процес мерења времена интервала између откуцаја на тастатури, битска неодређеност се смањује и могу се деградирати претпостављене случајне карактеристике извора, те излазни низ учинити предвидивим.

Други начин се базира на коришћењу специјализованог **хардвера за генерисање случајних бита**. Ови уређаји су базирани на природним појавама за које се сматра да се одвијају на случајан начин. Један од примера „непредвидивости“ је термални шум полупроводничке диоде. Случајност ових процеса се често усложњава комбиновањем више случајних извора.

За сигурност система на бази *RSA* алгоритма најбитније је очување тајности параметра d . Како је један од начина да се дође до тајног експонента d факторизације броја n , мора се водити рачуна да се избором бројева p и q не олакша могућност факторизације. *RSA* алгоритам се показао прилично робусним и користи се у већини савремених система заштите.

3.3. Дигитална енvelope

Дигитална енvelope (*Digital Envelope*) представља комбинацију симетричних и асиметричних криптографских алгоритама којом је решен проблем дистрибуције кључева, карактеристичан за симетричне криптографске системе [13].

У процесу креирања дигиталне енvelope врши се шифровање садржаја поруке применом одређеног симетричног алгоритма (*DES*, *3DES*, *RC2*, *RC4*, *IDEA*, *AES* или неког наменског тајног алгоритма). Кључ симетричног алгоритма се шифрује јавним кључем примаоца поруке, применом одговарајућег асиметричног алгоритма. Шифрован садржај поруке са симетричном криптографском трансформацијом и шифрован симетрични тајни кључ са асиметричним алгоритмом представљају дигиталну енvelope. Кључ којим је шифрована порука може бити дешифрован једино применом тајног кључа субјекта коме је порука упућена.

Кључеви симетричних криптографских алгоритама могу се чувати на спољним меморијским уређајима (нпр. смарт картице), што захтева коришћење централизованог система управљања кључевима и системом заштите.

3.3.1. Сесијски симетрични кључеви

Алтернативни систем управљања кључевима јесте коришћење сесијских симетричних кључева. При свакој успостави везе између учесника у комуникацији, генеришу се случајне величине, што представља тајни кључ одговарајућег симетричног

алгоритма (сесијски кључ). На тај начин се користе различити симетрични кључеви, тако да их није потребно чувати на спољним меморијским уређајима [14].

Сесијски кључеви се најчешће креирају применом генератора псеудослучајних бројева, што смањује њихов степен безбедности, те су мање безбедни од централизованих система за управљање кључевима. Применом сесијског кључа реализује се процес криптографске трансформације порука које се преносе ка пријемној страни. Шифрована порука заједно са сесијским кључем, применом механизма дигиталне енvelope, шаље се пријемној страни. Дешифровање сесијског кључа може да изврши једино корисник коме је послата порука, применом операције приватног кључа одговарајућег асиметричног алгоритма.

Могуће је реализовати комбиновани систем управљања кључевима код којих се симетрични кључ добија комбинацијом кључа са спољног меморијског уређаја (смарт картице) и сесијског кључа. Наведеном комбинацијом кључева добија се „кључ поруке“ (*message key*).

3.4. Криптографске компресионе функције

Криптографске компресионе функције су познате и као једносмерне *hash* или *message digest* функције. Циљ ових функција је у одређивању отиска одређене поруке, односно провере интегритета података [13].

Најједноставнији пример употребе *hash* функције је одређивање бита парности отиска поруке. Процедура употребе криптографске компресионе функције се огледа у упоређивању добијеног отиска поруке и поновно генерисаног отиска.

При употреби криптографских компресионих функција морају бити задовољени следеће критеријуми:

- могуће је креирање криптографског отиска поруке,
- немогуће је реконструисати изворну поруку на основу криптографског отиска и
- постоји занемарљива вероватноћа креирања идентичних криптографских отисака са истим карактеристикама за различите поруке.

Основна подела криптографских компресионих функција је:

- криптографске компресионе функције са употребом тајних кључева:
 - блок *MAC* и
 - секвенцијални *MAC*.
- криптографске компресионе функције без употребе тајних кључева:
 - *MD4* и *MD5* и
 - *RIPEMD-128* и *RIPEMD-160*.

Криптографске компресионе функције са применом тајног кључа називају се и функцијама за проверу аутентичности поруке (*MAC* - *Message Authentication Code*) пошто се, осим контроле интегритета порука, проверава и аутентичност пошиљаоца поруке. Оне се заснивају на употреби стандардних криптографских компресионих функција, уз примену тајних параметара.

Криптографске компресионе алгоритме без употребе тајног кључа *MD4* и *MD5* (*MD* - *Message Digest*) први је објавио амерички математичар *Ron Rivest*.

Алгоритам *SHA1* (*Secure Hash Algorithm*) развио је *NIST* у процесу дефинисања стандарда за дигитални потпис (*DSS - Digital Signature Standard*). У оквиру пројекта заштите Европске уније под називом *Race Integrity Primitives Evaluation (RIPE)*, развијен је алгоритам за креирање криптографског отиска поруке *RIPEMD (RIPE Message Digest)*.

3.4.1. MD5 алгоритам

Алгоритам *MD5* обрађује поруке максималне дужине 2^{64} -bit дајући као резултат криптографску компресиону вредност од 128-bit. У *MD5* алгоритму, након иницијализације параметара, обрађује се улазна порука у блоковима од 512-bit. Уколико битска дужина улазне поруке по модулу 512 није једнака 448, проширује се дата порука са најмањим бројем бита потребним за реализацију претходно наведеног захтева. Проширивање улазне поруке се реализује тако што се на крај дате поруке додаје један бит јединице, праћен потребним бројем бинарних нула. Након реализације претходно описаног процеса, врши се повезивање 64-bit репрезентације дужине улазне поруке на крај последњег блока. Наведеним поступком формира се порука чија је дужина умножак од 512-bit. Приликом обраде у *MD5* алгоритму, 512-bit блокови поруке деле се у шеснаест подблокова дужине 32-bit над којима се реализују све операције. Резултујућа вредност алгоритма представља скуп од четири 32-bit блока, који се повезују тако да једнозначно формирају 128-bit вредност отиска поруке [7] [9].

3.4.2. SHA1 алгоритам

SHA1 алгоритам процесира поруке у блоковима од 512-bit, подељених у 16 блокова од 32-bit. Поступак је сличан као и код *MD5* алгоритма. Порука се проширује до дужине која је за 64-bit краћа од 512-bit мултипла. На крају се додаје јединица праћена одговарајућим бројем нула. Потом се 64-bit репрезентација поруке додаје резултату. Излаз из алгоритма представља скуп од пет 32-bit блокова, тако да заједно формирају 160-bit hash вредност [7] [9].

4. Технологије заштите информационих система

4.1. Инфраструктура јавног кључа

Инфраструктура јавног кључа (*PKI - Public Key Infrastructure*) омогућава поуздано успостављање идентитета субјеката у облику дигиталних сертификата. На тај начин се, као спрега хардверских и софтверских елемената, ствара окружење у коме је према прецизно утврђеној политици рада могуће реализовати следеће безбедносне сервисе [15]:

- **заштита тајности** - реализује се симетричним шифарским системима,
- **аутентичност** - реализује се асиметричним шифарским системима,
- **интегритет података** - реализује се асиметричним шифарским системима и
- **непорецивост трансакција** - реализује се асиметричним шифарским системима.

Функционалне целине инфраструктуре јавног кључа треба да обезбеде:

- поверење у једнозначну везу између идентитета субјекта и јавног кључа,
- висок криптолошки квалитет јавних и приватних кључева за асиметричне криптографске алгоритме и тајних кључева за симетричне криптографске алгоритме и
- највиши степен тајности генерисаних кључева.

Примери апликација заснованих на инфраструктури јавног кључа су:

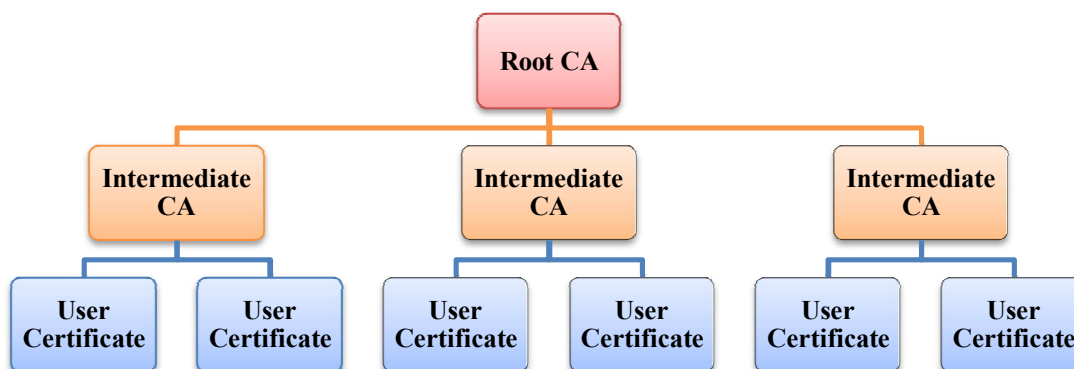
- заштита софтверских апликација и мрежних комуникација,
- формирање виртуелне приватне мреже (*VPN - Virtual Private Network*),
- формирање система заштите на транспортном нивоу у рачунарској мрежи,
- системи електронског плаћања,
- заштићен *e-mail* сервис и слање порука,
- безбедна размена докумената итд.

Инфраструктура јавног кључа садржи следеће компоненте:

- **сертификациони ауторитет** (*CA - Certification Authority*) који издаје дигиталне сертификате и регулише начин њихове употребе,
- **регистрациони ауторитет** (*RA - Registration Authority*) као инстанца за подношење захтева за издавање сертификата,
- **комуникациони систем** за дистрибуцију захтева за издавање сертификата и слање дигиталних сертификата између регистрационог и сертификационог ауторитета,
- **криптографске апликације** за реализацију функција *PKI* и
- **субјекте** у мрежном окружењу.

У систему са јавним кључевима потребно је да постоје:

- **Root CA** - сертификационо тело од највишег поверења задужено за издавање дигиталних сертификата другим сертификационим телима (*Intermediate CA*),
- **Intermediate CA** - сертификационо тело коме дигитални сертификат издаје *Root CA* и које издаје дигиталне сертификате за крајње кориснике (*User Certificate*) и
- **User Certificate** - сертификати крајњих корисника у *PKI* којима сертификате издају *Intermediate CA*.



Шема 4-1 Хијерархијска структура сертификационих тела

PKI системи су кључна карика савремених информационих система и сигурно ће доминирати у безбедносним системима будућности.

4.2. Сертификационо тело и дигитални сертификат

Сертификациони ауторитет је тело са највећим степеном поверења у *PKI* систему. Он представља језгро читавог *PKI* система. *CA* својим дигиталним потписом гарантује асоцијацију одговарајућег субјекта и његовог јавног кључа [16].

Основне функције *CA* су:

- генерисање и издавање дигиталних сертификата,
- управљање роком важности и
- управљање процедуром повлачења сертификата.

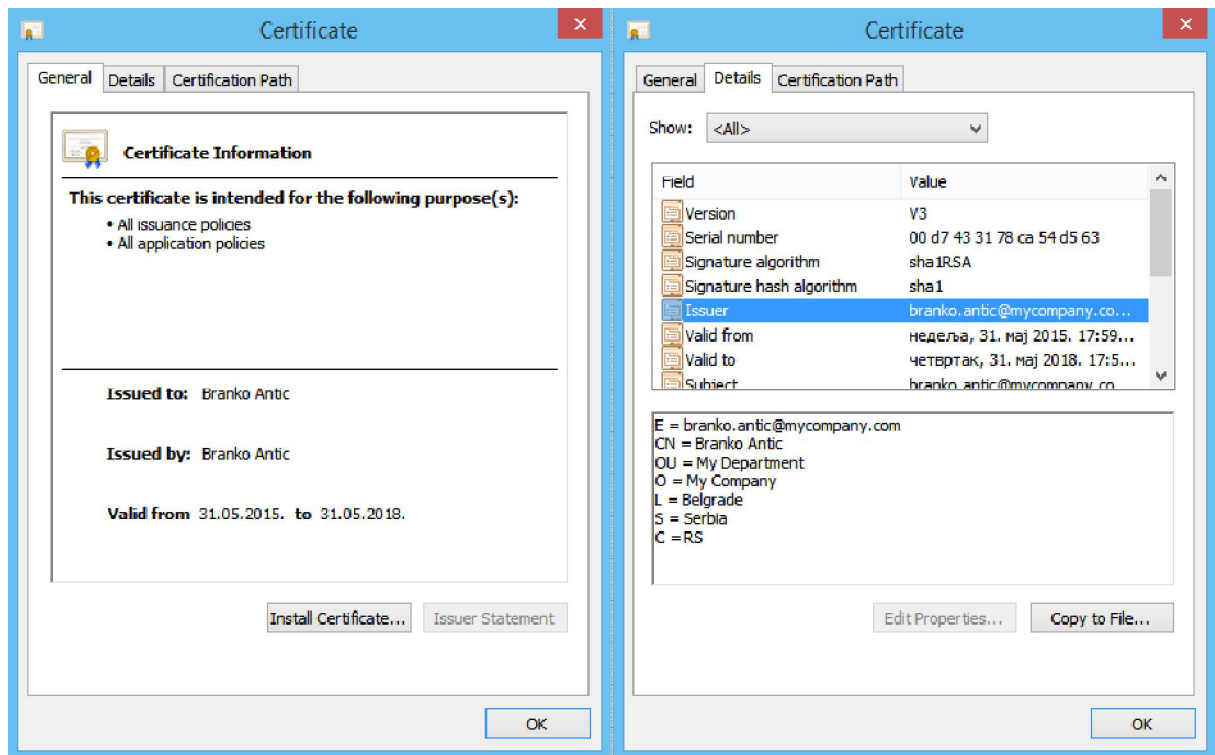
У савременој криптографији генерисање кључева и креирање дигиталног сертификата на бази јавног кључа реализује се на два начина.

Сертификационо тело генерише пар јавног и приватног кључа, формира дигитални сертификат и доставља приватни кључ и сертификат власнику. Завршетком процеса креирања и издавања сертификата, корисник постаје власник дигиталног сертификата. Сертификационо тело својим дигиталним потписом, на бази асиметричног криптографског алгорита, гарантује везу између јавног кључа и идентитета власника сертификата. Сертификационо тело, као страна од поверења, може контролисати и одржати јединствен квалитет креираних кључева и јединственост процедуре безбедног чувања кључева.

У савременим *PKI* системима, кључеви се генеришу и испоручују субјектима, најчешће, као садржај заштићене меморије смарт картице. Генерисање пара јавног и приватног кључа реализује корисник применом хардверских или софтверских механизма, након чега се врши креирање захтева за издавањем сертификата (*certificate request*), који садржи јавни кључ корисника. Предност наведеног начина јесте у томе да се приватни кључ асиметричног алгорита налази само на једној локацији у систему, која припада власнику датог кључа.

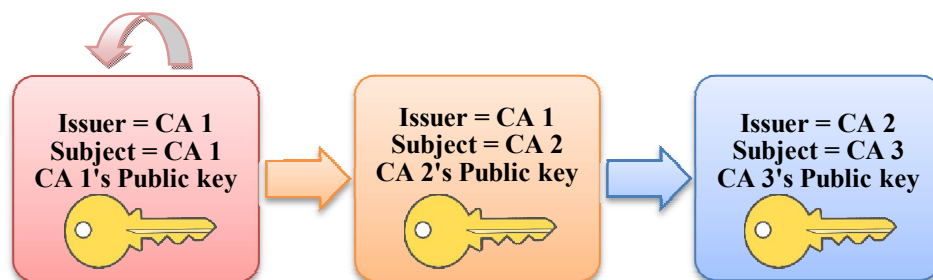
У процесу утврђивања идентитета учесника у комуникацији, на бази размене дигиталних сертификата, сертификационо тело представља трећу страну од поверења (*TTP - Trusted Third Party*), који својим дигиталним потписом гарантује исправност дигиталног сертификата учесника.

Дигитални сертификат (*Digital Certificate*) представља елемент којим се утврђује веза између идентитета субјекта и његовог јавног кључа примењеног алгоритма [17].



Слика 4-2 Дигитални сертификат

У самом сертификату се налази и податак о ланцу сертификационих тела. Сертификат сваког сертификационог тела је издат од стране надређеног сертификационог тела, чиме се образује ланац сертификата. На крају ланца се налази корени (самопотписани) сертификат, који је потписан сопственим приватним кључем.



Шема 4-3 Ланац дигиталних сертификата

4.3. Дигитални потпис

Дигитални потпис (*Digital Signature*) је скуп података у електронском облику обележених потписниковим тајним параметром, који су додати или логички придружени електронским порукама или документима [5].

Сврха дигиталног потписа је да потврди аутентичност садржаја поруке (доказ да порука није промењена на путу од пошиљаоца до примаоца), као и да обезбеди гарантовање идентитета пошиљаоца поруке, тј. идентификује потписника [18].

Реализација поступка дигиталног потписа значајна је за апликације које захтевају међусобну проверу аутентичности страна у комуникацији, као и заштиту интегритета података који се преносе кроз дистрибуирани рачунарски систем [6].

Постоје две основне категорије механизма дигиталног потписа:

Механизам дигиталног потписа са додатом изворном поруком (*Digital Signature Schemes With Appendix*) уз дигитални потпис шаље и оригиналну поруку да би пријемна страна могла извршити процес верификације. Овај механизам се може реализовати применом алгоритама: *RSA*, *DSA* (*Digital Signature Algorithm*), *ECDSA* (*Elliptic Curve DSA*), *ElGamal*, *Schnorr* итд.

Механизам дигиталног потписа са реконструкцијом изворне поруке (*Digital Signature Schemes With Message Recovery*), у процесу верификације дигиталног потписа, врши обнављање информационог садржаја криптографски заштићене поруке. Наведени механизам се користи код порука мале битске дужине и најчешће се реализује применом алгоритама: *RSA*, *Rabin*, *Nyberg-Rueppel* итд.

За креирање криптографског отиска поруке најчешће се користе алгоритми *MD5* и *SHA1*. За реализацију шифарске трансформације у поступку креирања и верификације дигиталног потписа, углавном се користе алгоритми *RSA*, *DSA* и *ECDSA*.

4.4. Креирање и верификација дигиталног потписа

Процедура креирања дигиталног потписа поруке садржи следеће фазе:

- одговарајућом криптографском компресионом функцијом се редукује садржај поруке и креира отисак поруке,
- креирани отисак поруке се шифрује применом асиметричног криптографског алгоритма и приватним кључем корисника,
- шифровани отисак поруке, који представља дигитални потпис, придружује се поруци и шаље пријемној страни.

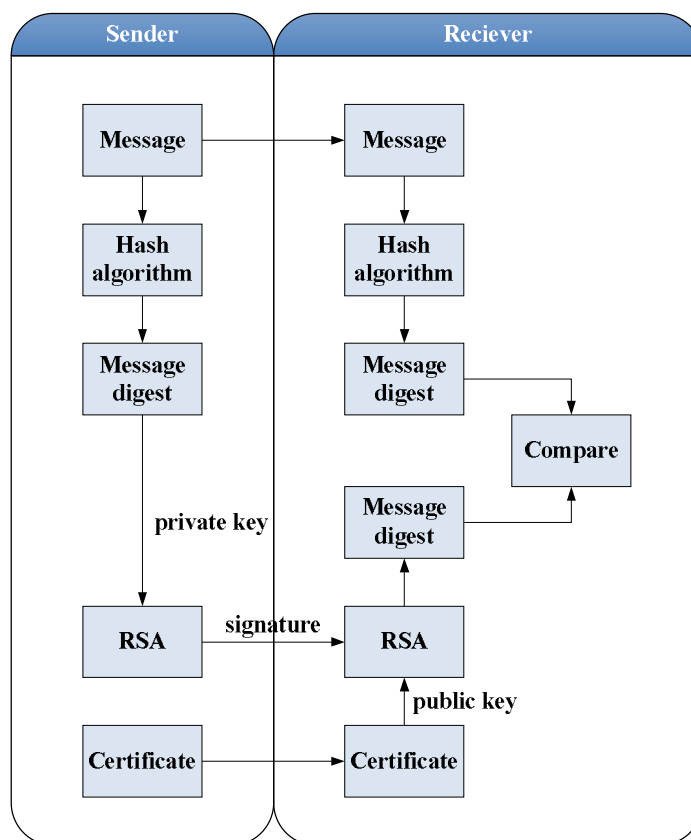
Да би се могао извршити процес верификације потписа дигиталног сертификата, неопходно је да јавни кључ (*Public key*) асиметричног криптографског алгоритма сертификационог тела буде познат свим учесницима у комуникацији.

Процедура верификације дигиталног потписа садржи следеће фазе:

- из добијене поруке се издваја дигитални потпис и дешифрује се јавним кључем,
- поново се израчунава криптографски отисак поруке идентичним поступком као на предајној страни,
- уколико је креирани отисак идентичан са дешифрованим примљеним отиском, верификација дигиталног потписа је успешна.

Успешном провером дигиталног потписа потврђене су следеће карактеристике безбедности система:

- **аутентичност** - отисак поруке који је шифрован корисниковим тајним кључем, могуће је дешифровати само одговарајућим јавним кључем,
- **интегритет поруке** - идентични су израчунати и дешифровани отисак поруке и
- **непорецивост** - порука је потписана тајним кључем пошилаоца.



Дијаграм 4-4 Поступак креирања и верификације дигиталног потписа

4.5. Захтеви за сертификате и листе опозваних сертификата

Да би се доделио дигитални сертификат, неопходно је да се креира **захтев за добијање сертификата** (*Certificate Request*). Он се доставља одговарајућем сертификационом телу и садржи податке о кориснику. Захтев за сертификат је дигитално потписан у циљу гаранције његовог интегритета. Сертификационо тело верификује захтев коришћењем јавног кључа који је у оквиру захтева достављен [19].

Током генерисања дигиталних сертификата, сертификационо тело креира и **листе опозваних сертификата** (*CRL*). Листом опозваних сертификата се врши провера валидности дигиталних сертификата учесника у комуникацији. Такође листа је дигитално потписана од стране сертификационог тела, тако да су корисници сигурни у аутентичност и интегритет информација датих у листи.

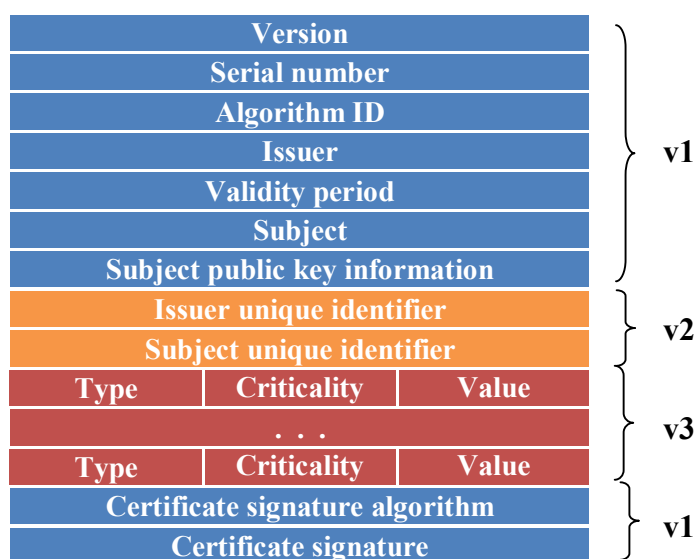
Скуп поља и података које листа опозваних сертификата треба да садржи, дефинисана је стандардом *X.509* верзија 2 (*RFC - Request for Certificate 2459*) [20], а то су:

- верзија формата листе опозваних сертификата - ознака структуре *CRL* по стандарду *X.509* (верзије 1 и 2),
- назив издавача *CRL*,
- време објављивања *CRL*,
- време када ће сертификациони ауторитет најкасније објавити следећу, ажурирану верзију *CRL*,
- попис опозваних сертификата према серијским бројевима, са разлогом њиховог повлачења,

- идентификатор додатних идентификационих поља као што су: редни број листе, дистрибуционе тачке листе итд,
- идентификатор алгоритма примењеног за дигитално потписивање дате *CRL* (*RSA* са *MD5*, *RSA* са *SHA-1*, *DSA* итд.), и
- дигитални потпис *CRL*.

4.6. X.509 стандард дигиталних сертификата

Стандард **X.509** представља стандард из групе *X.500* стандарда, а који прописују структуру, поступак добијања и начин представљања сертификата. Објављени су 1988. године, а 1993. и 1996. године су објављене верзије 2 и 3 стандарда *X.509* [19].



Шема 4-5 Структура дигиталног сертификата према *X.509* стандарду

Version поље садржи информацију о верзији формата *X.509* сертификата.

Serial number поље садржи серијски број сертификата, који представља јединствену ознаку сваког сертификата у оквиру домена одређеног сертификационог тела. Серијски број има улогу при опозиву сертификата, јер се на тај начин идентификује на листи опозваних сертификата.

Algorithm ID поље одређује алгоритам за потписивање сертификата као и *hash* алгоритам, према идентификационом броју, нпр. *RSA* и *MD5*. Алгоритам потписа је битан због провере валидности дигиталног потписа.

Issuer поље садржи јединствено име сертификационог тела које је издало сертификат (*DN - distinguished name*). Састоји се од следећих поља:

- *CN (Common Name)* - име власника,
- *O (Organization)* - назив организације,
- *OU (Organization Unit)* - назив ниже организацијске јединице,
- *L (Location)* - назив места издавања,
- *C (Country)* - двословна ознака државе и
- *S (State)* - регион у оквиру државе.

Validity period поље одређује временски интервал валидности сертификата. Сваки пут када се сертификат употреби проверава се је ли сертификат унутар рока валидности.

Subject пољем одређује се *X.500 DN* ентитета који је власник приватног кључа за који се издаје сертификат.

Subject public key information садржи:

- јавни кључ сертификата и
- идентификациони број алгоритма који се користи уз јавни кључ и *hash* алгоритам.

Certificate authority signature гарантује да је сертификат дигитално потписан од стране сертификационог тела које га је издало.

Верзијом 2 стандарда *X.509* уведена су поља *Issuer unique identifier* и *Subject unique identifier*.

Issuer unique identifier је поље уведено из разлога идентификације *Issuer X.500 name*, у случају да је више сертификационих тела имало исти *X.500 DN*.

Subject unique identifier је поље уведено за потребе јединствене идентификације *Subject X.500 name*.

Верзијом 3 стандарда *X.509* омогућен је механизам проширивања сертификата додатним пољима. Унапред су дефинисана „стандардна проширења“, али се сертификати могу проширити и другим пољима по одобрењу одговарајућих међународно признатих организација за стандардизацију, као што је *ISO*.

Свако проширење се састоји од три поља:

- *type* - тип,
- *criticality* - критичност и
- *value* - вредност.

Type поље одређује тип податка поља *value*. Тип може бити текст, број, датум, графика или нека сложена структура података.

Criticality поље је ознака критичности поља *value*. На тај начин се дефинише да ли се податак у пољу *value* сме занемарити. Ако се критично проширење не може обрадити, сертификат се мора одбацити.

Value поље садржи податке.

Стандардна проширења сертификата подељена су у следеће групе:

- информације о кључу - информације о намени сертификата и пара кључева,
- информације о политици (*policy*) - начин на који се одређени сертификат мора користити и интерпретирати.
- атрибути корисника и сертификационог тела (*basic constraints*) - информације за идентификацију корисника и сертификационог тела и
- ограничења на путањи сертификације - механизми којима сертификационо тело управља и ограничава поверење „проширено“ на треће особе.

Критична поља се користе за податке веома велике важности, да га морају разумети све апликације. Велика већина проширења је некритична. Неопходан је опрез и детаљна анализа приликом додавања критичних проширења, јер могу проузроковати проблеме при коришћењу сертификата. Насупрот критичним пољима, одређено проширење може бити нужно апликацији, али то не значи да такво поље мора бити означено као критично.

4.7. Стандарди криптографије јавног кључа

Стандарди криптографије јавног кључа (*PKCS - Public Key Cryptography Standards*) дефинишу интероперабилне механизме за потребе реализације *PKI* система [21].

Стандард	Домен
<i>PKCS #1</i>	стандард шифровања и потписивања података који дефинише криптографске елементе, начине шифровања и дешифровања и начине потписивања и верификовања потписа
<i>PKCS #2</i>	стандард је инкорпориран у стандард <i>PKCS #1</i>
<i>PKCS #3</i>	стандард протокола за размену кључева
<i>PKCS #4</i>	стандард је инкорпориран у стандард <i>PKCS #1</i>
<i>PKCS #5</i>	стандард шифровања и дешифровања тајним кључем генерисаним помоћу лозинке
<i>PKCS #6</i>	стандард структуре дигиталног сертификата - <i>X.509</i> в3
<i>PKCS #7</i>	стандард синтаксе шифроване поруке и ентитета (листа повучених сертификата, индентификатор алгоритма, дигитални сертификат, издавач сертификата, верзија дигиталног сертификата итд.)
<i>PKCS #8</i>	стандард синтакси података о тајном кључу и метода чувања тајног кључа
<i>PKCS #9</i>	стандард типова атрибута других <i>PKCS</i> стандарда
<i>PKCS #10</i>	стандард синтакси захтева за сертификате
<i>PKCS #11</i>	стандард интерфејса криптографског токена (<i>Cryptoki</i>) за приступ уређајима са подршком за криптографску технологију, као што су смарт картице, хардверски безбедносни модули, <i>PCMCIA</i> картице итд.
<i>PKCS #12</i>	стандард синтакси за размену података, кључева, сертификата итд.
<i>PKCS #13</i>	стандард механизма шифровања и потписивања података криптографским системима елиптичких крива
<i>PKCS #14</i>	стандард генератора псеудослучајних бројева
<i>PKCS #15</i>	стандард информација на криптографском токenu

Табела 4-6 *PKCS* стандарди

4.8. Протоколи

4.8.1. SSL протокол

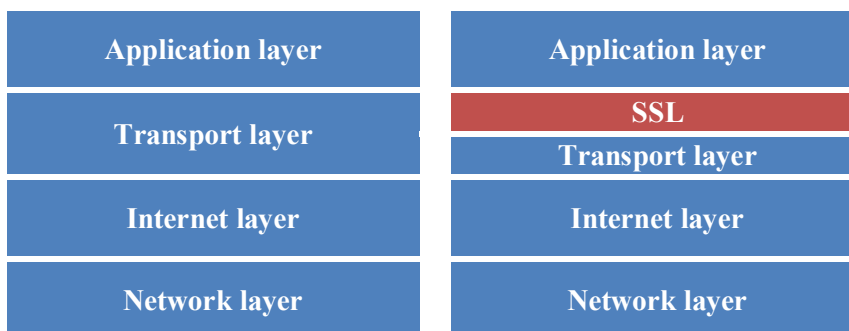
SSL (*Secure Sockets Layer*) је протокол заштите на транспортном нивоу, опште намене, који омогућава заштиту тајности у комуникацији клијент-сервер програма. Протокол омогућава клијент - сервер апликацијама да комуницирају на начин који спречава прислушкивање и промену садржаја порука. Представља један од интернет стандарда [22].

Прву верзију овог протокола је предложила компанија *Netscape*, затим је широко прихваћен и имплементиран у већини савремених *web* претраживача. Извршено је неколико ревизија *SSL* протокола. Актуелна верзија 3.0 је доживела измене кроз спецификацију *TLS* (*Transport Layer Security*) протокола.

Основни циљ *SSL* протокола је да обезбеди тајност и поузданост података који се размењују између програма. Протокол се састоји од два нивоа:

- Нижи ниво *SSL* протокола представља *SSL Record* протокол, који користи поуздани мрежни протокол (*TCP - Transmission Control Protocol*).
- Виши ниво *SSL* протокола је *SSL handshake* протокол којим се врши узајамна провера аутентичности клијента и сервера, уговарање криптографског алгорита који ће се користити у комуникацији и безбедна размена кључева.

SSL је протокол опште намене, јер се у референтном *TCP/IP* (*Transmission Control Protocol / Internet Protocol*) мрежном моделу, налази одмах изнад транспортног слоја, а испод апликативног. Овим се постиже транспарентност *SSL* протокола у односу на сервисе апликативног слоја. Самим тим, сви сервиси апликативног нивоа који користе *TCP* протокол, могу користити *SSL* протокол, остварујући на тај начин заштиту података које размењују.



Шема 4-7 *TCP/IP* мрежни модел и *SSL* протокол

Основна апстракција на којој се заснива *SSL* протокол је конекција, што је директно условљено чињеницом да се најчешће заснива на поузданом *TCP* протоколу. Другим речима, *SSL* конекција енкапулира *TCP* конекцију додајући јој нова својства: тајност размењених података и проверу њиховог интегритета.

SSL протокол обезбеђује сигурност конекција са три аспекта:

- Конекције су тајне. Након иницијалног „руковања“ (*handshake*) дефинише се кључ за уговорени симетрични криптографски алгоритам који се затим користи за шифровање порука.

- Провера аутентичности страна у комуникацији се врши применом асиметричних криптографских алгоритама.
- Конекција је поуздана. Размена порука укључује проверу интегритета заснованог на *MAC* функцијама.

4.8.2. S/MIME протокол

Данас, вероватно један од најкоришћенијих протокола на апликативном нивоу је **S/MIME** (*Secure / Multipurpose Internet Mail Extension*). *S/MIME* апликације се уграђују у софтверске пакете који су данас најдоминантнији на тржишту, од којих су неки: *Microsoft Outlook*, *Mozilla Firefox* итд.

S/MIME је базиран на популарном интернет *MIME* стандарду и он обезбеђује следеће криптографске услуге по питању сигурности за апликације типа електронске размене порука: аутентификација, интегритет поруке и непорецивост (користећи дигитални потпис) и тајност података (дигиталну енVELOпу) [23].

S/MIME може бити коришћен од стране традиционалних корисничких поштанских агената (*MUA - Mail User Agents*) како би се пошти која је послата додале криптографске безбедносне услуге и како би се интерпретирале криптографске безбедносне услуге у пошти која је примљена.

Међутим, *S/MIME* није ограничен само на електронску пошту; може се користити са било којим транспортним механизмом који транспортује *MIME* податке, као што је *HTTP (HyperText Transfer Protocol)*. Осим тога, *S/MIME* може бити примењен у агентима аутоматизованог преноса порука који користе криптографске безбедносне услуге које не захтевају било какву интервенцију човека, као што је потписивање софтверски-генерисаних докумената и енкрипцију факс порука које се шаљу преко интернета.

5. Смарт картице

Смарт картице представљају носиоце дигиталног сертификата. Имају примену у телекомуникацијама као *SIM (Subscriber Identity Module)* картице, у здравству као здравствена књижица, у финансијским трансакцијама као платне картице, као лична карта, идентификациона картица, „електронска карта“ у јавном превозу итд. Подразумевани су облик личних легитимација свуда у свету. Висок ниво флексибилности смарт картица у проширењу функционалности, перманентно добија на ширини дијапазона примене, који превазилази границе традиционалне примене картица. Практично, могућности употребе смарт картица су неограничене [24].

5.1. Историјат смарт картица

Развој смарт картица започет је средином XX века. Пионир у овој области је *Dinners Club* који је 1950. године издао прву пластичну картицу, која је у том тренутку била само статусни симбол [25].



Слика 5-1 Смарт картица

У свом зачетку смарт картице су се издавале у облику рељефних картица. Нису имале никакав вид заштите и углавном су служиле за идентификацију. Њихова основна функција се заснивала на складиштењу података, као што су име власника, које је најчешће било одштампано на површини картице, док је број картице био утиснут на њој. Сигурност картица се заснивала пре свега на савести људи који су их користили што се испоставило као мана, јер су картице ускоро могли имати скоро сви.

Прве картице као носиоци информација имале су картице са магнетним тракама. То је омогућавало да се подаци у дигиталном облику сместе на картицу, што је условило нове методе идентификације корисника. Уместо потписа уведено је коришћење персоналних идентификационих бројева (*PIN - Personal Identification Number*). Кључна слабост ових картица, поред малог капацитета за смештај података, била је да се подаци могу лако прочитати и изменити. Због тога се нису могле користити за чување поверљивих података. Компромис је био у сталној конекцији терминала и рачунара на коме су биле похрањени *PIN*-ови картица, за потребе упоређивања референтне вредности *PIN*-а са *PIN*-ом који уноси корисник, упркос повећању трошкова функционисања система.

Развој микроелектронике и силиконских чипова 70-их година XX века омогућио је нове трендове у развоју картица. На чипу су могли бити смештени подаци и логика за обраду података. Наведену технологију су *Jürgen Dethloff* и *Helmut Grötrupp*

патентирали у виду идентификационих картица. Развој индустрије полупроводника и производња интегрисаних кола по комерцијалним ценама, омогућио је да *Roland Moreno* патентира смарт картице у Француској 1974. године.

Велики пробој у области смарт картица, упркос дотадашњим применама десио се у области телекомуникација 1984. године, пробним пројектом телефонских картица француског ПТТ. Постигнута су сва очекивања у погледу високе поузданости и заштите од неауторизоване манипулације подацима.

У Немачкој је 1984. године покренут пројекат упоредног тестирања картица са магнетном траком, оптичких картица и смарт картица у улози телефонских картица. Са оствареним највећим степеном флексибилности, поузданости и сигурности од неауторизованог коришћења података, смарт картице су однеле победу.

За разлику од интегрисаних кола у телефонским картицама који су релативно мали, једноставни и јефтини меморијски чипови са специфичном логиком за заштиту, микропроцесорски чипови су значајно већи и комплекснији. Најпре су коришћени у мобилним телекомуникацијама. *German Post Office* је 1988. године увео модерне микропроцесорске картице које користе *EEPROM* (*Electrically Erasable Programmable Read-Only Memory*) технологију као картице за ауторизацију у аналогној мобилној телефонској мрежи (*C-Netz*). Из техничких разлога, аналогне мобилне телефонске мреже су ограничене на релативно мали број претплатника (око једног милиона), те не постоји право масовно тржиште за микропроцесорске картице. Ипак, позитивна искуства стечена приликом коришћења картица у аналогним мобилним телефонским мрежама су била одлучна за улазак смарт картица у дигиталне *GSM* (*Global System for Mobile Communications*) мреже. Такве мреже су пуштене у рад 1991. у неколико европских земаља, а данас су распрострањене у свим земљама света.

Развој смарт картица, криптографије и полупроводничке технологије, као и целокупно напредовање у области хардвера и софтвера, омогућили су да се користе комплексни математички алгоритми у циљу обезбеђења заштитних механизма електронским системима за обраду података. Технологија је постала свима доступна, за разлику од периода када је била примењивана само у војним и тајним службама.

Смарт картице у области финансија и трговине уводе *Visa* и *MasterCard*. Први систем коришћења смарт картица за финансијске трансакције је назван *Danmont* и пуштен у рад у Данској 1992. године. У 1997. све немачке асоцијације за штедњу и велики број банака почеле су да користе смарт картице. Применом мултифункционалних смарт картица са *POS* (*Point of Sales*) функцијама, електронским новцем и опционим додатним услугама, Аустрија је постала прва земља у свету у којој је у целости био развијен систем електронског плаћања. Данас, платне картице омогућују људима да купују било где у свету. Та јединствена предност је помогла да се смарт картице брзо распростране по свету, а плаћање преко интернета представља још један сегмент глобалне појаве електронског пословања.

5.2. Карактеристике и типови смарт картица

Смарт картице су постале идеалан медијум за сврхе у којима се користе због својих основних карактеристика.

Интегрисано коло уграђено у смарт картицу има **могућности преноса, чувања и обраде података**.

Картице обезбеђују **висок ниво сигурности** података доступан свима, с обзиром да могу извршавати криптографске алгоритме и чувати тајне кључеве. С обзиром на спрегу хардверско-софтверских решења која се користе за комуникацију са смарт картицама, подаци се могу уписати и чувати на картици тако да буду безбедни од читања ван картице, тј. од неауторизованог приступа подацима. То омогућава креирање услова под којима ће се вршити уписивање, брисање, читање и чување података, а самим тим и конструисање разних видова механизма за заштиту, прилагодљивих специфичним потребама. У комбинацији са могућношћу употребе криптографских алгоритама, смарт картице могу се искористити за имплементацију погодних сигурносних модула [25].

Картице су **једноставне за коришћење**, јер због својих малих димензија и једноставности употребе могу бити ношене и коришћене било где и може их користити било ко у свакодневном животу.

Ниска цена *PVC (Polyvinyl Chloride)* материјала од које се праве картице, омогућила је производњу робусних, дуготрајних пластичних картица, подесних за свакодневну употребу и **отпорних** на механичке и климатске услове, као и отпорност на хабање.

Максимални **капацитет** смарт картице је неколико пута већи од капацитета картице са магнетном траком. Такође, **поузданост и рок трајање** смарт картица је неупоредив са магнетном траком.

Основне карактеристике и функционалности смарт картица су дефинисане *ISO-7816* стандардом [24].

Смарт картице могу се поделити на:

- меморијске картице и
- микропроцесорске картице.

Такође могу се поделити и на:

- контактне картице и
- бесконтактне картице.

5.2.1. Меморијске картице

Меморијске картице немају микропроцесор, већ меморијски чип или чип са логиком и меморијом, најчешће капацитета од *1KB* до *4KB*. Најчешће су коришћене као *prepaid* картице за јавне телефонске говорнице.

Интегрисани чип у облику једноставног логичког кола може да изврши само унапред програмиране инструкције, које не могу бити репрограмиране. На тај начин су коришћене у једнократне сврхе, након чега се више не могу користити.

Сигурност меморијских картица може бити имплементирана у смислу заштите меморије или тајне логике. На примеру *prepaid* телефонских картица, програмира се

логика која онемогућује повећање вредности на картици. Међутим, могуће је релативно једноставно фалсификовати меморијску картицу.

Ниска цена израде меморијских картица и једнократност употребе, довољни су услови за њихову примену у областима где је цена битан фактор и не постоје велики ризици злоупотребе.

5.2.2. Микропроцесорске картице

Као што и назив ових врста картица говори, микропроцесорске картице поседују микропроцесор. Ограничења у функционалности су дефинисана брзином рада микропроцесора и величином меморијског простора [25].

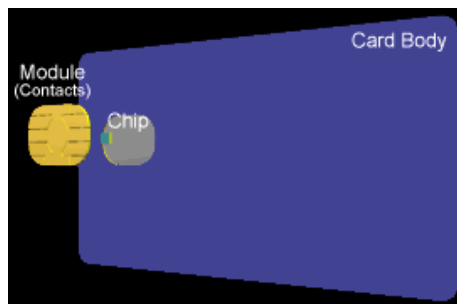
Из угла безбедности, оне имају повећани ниво сигурности у односу на меморијске картице. Подаци нису директно доступни спољашњим апликацијама, већ микропроцесор контролише приступ и коришћење меморије према дефинисаним условима као што су лозинка, енкрипција итд. Савремене микропроцесорске картице имају подршку за криптографију и користе се у апликацијама где се захтева заштита података.

Још једна карактеристика картица је флексибилност у погледу могућности различитих примена. Микропроцесор им омогућава да буду оптимизоване за једну или више примена.

Данас се микропроцесорске картице масовно производе по комерцијалним ценама, која варира у зависности од меморијског капацитета и софтерских могућности. Најчешћа примена је у системима за контролу приступа, у електронској трговини и телекомуникацијама. Поред наведеног, микропроцесорске картице се примењују и свуда где је главни циљ сигурност и безбедност података и трансакција.

5.2.3. Контактне картице

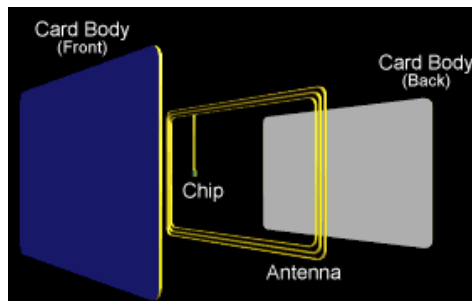
Контактне картице представљају картице са којима се комуникација остварује преко конекције модула интегрисаног на картици и уређаја за читавање - терминала или читача картица [25].



Слика 5-2 Контактна смарт картица

5.2.4. Бесконтактне смарт картице

Бесконтактне картице, за разлику од контактних, имају могућност комуникације са уређајима без директног механичког спајања. Конекција се остварује помоћу интегрисане антене која врши примопредају преко електромагнетног поља [25].



Слика 5-3 Бесконтактна смарт картица

Ова врста картица се примењује у системима где се захтева брза трансакција, а то су најчешће системи контроле приступа и јавног превоза.

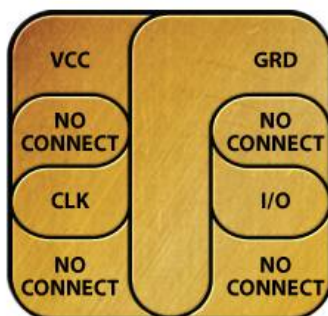
Бесконтактне картице превазилазе ограничења контактних картица - не постоје контакти који би се могли истрошити услед коришћења, не мора се водити рачуна о положају картице приликом убацивања у *CAD* (*Card Accepting Device*) уређаје и картице не морају имати одређени облик у складу са улазом *CAD* уређаја.

Основне мане бесконтактних картица је да морају бити у опсегу за комуникацију са *CAD* уређајима, ограничена је количина података у размени и могу да се врше трансакције без знања корисника картице. Као битан фактор у употреби бесконтактних картица је виша цена израде од контактних картица.

5.3. Хардверске карактеристике смарт картица

5.3.1. Електронски контакт чипа смарт картице

Комуникација са контактним картицама се остварује користећи серијски интерфејс за комуникацију. Конекција се остварује преко осам контактних тачака. Димензије и њихова локација је дата у другом делу стандарда *ISO-7816* [24].



Шема 5-4 Електронски контакти чипа смарт картице

- V_{cc} тачка обезбеђује напајање чипа. V_{cc} напон је 3V или 5V, са максималним одступањем од 10%.
- CLK тачка се користи за слање сигнала који ресетује микропроцесор – он се обично назива „топли ресет“. „Хладни ресет“ настаје када се напајање искључи и поново укључи. Вађење картице изван CAD уређаја и његово поновно убацивање проузрокује хладни ресет. Процесор на смарт картици не поседује свој интерни сигнал такта. CLK тачка прихвата екстерни сигнал такта од којег се прави интерни сигнал такта.
- GND тачка се користи као референтни напон; за чију вредност се подразумева да је 0V.
- I/O тачка се користи за пренос података и команди између смарт картице и спољашњег света у *half-duplex* моду рада.
- V_{pp} тачка је опциона и користила се у старијим типовима картица за програмирање *EEPROM* меморије. Обезбеђује програмирани напон са два нивоа. Нижи напонски ниво се назива пасивно стање; оно мора бити одржавано од стране CAD уређаја све док се не буде захтевао други напонски ниво - активно стање.

5.3.2. Микропроцесор смарт картице

Процесори који се користе у смарт картицама нису специјално дизајнирани за њих, него су то уређаји који су практично проверени употребом у другим областима. У индустрији смарт картица није уобичајено развијати нове процесоре који би се користили у посебним областима, јер је то прескупо. Уз наведено, било би потребно обезбедити и одговарајуће библиотеке функција и алате за развој, који би били доступни произвођачима оперативних система [25].

Процесори у већини садашњих чипова у смарт картицама имају *CISC (Complex Instruction Set Computer)* архитектуру. Најчешће је то 16-bit микроконтролер, који обично користи инструкцијски скуп *Motorola 6805* или *Intel 8051* процесора. Користе се и процесори који су засновани на *Intel 8051* архитектури као што су *Renesas H8*, *Philips XA* или *Samsung CALM*.

Најновији чипови у смарт картицама имају 32-bit микроконтролере. Доступне су и смарт картице са *RISC (Reduced Instruction Set Computer)* архитектуром. Један од познатих *RISC* процесора је и *Renesans H8*.

5.3.3. Копроцесори смарт картице

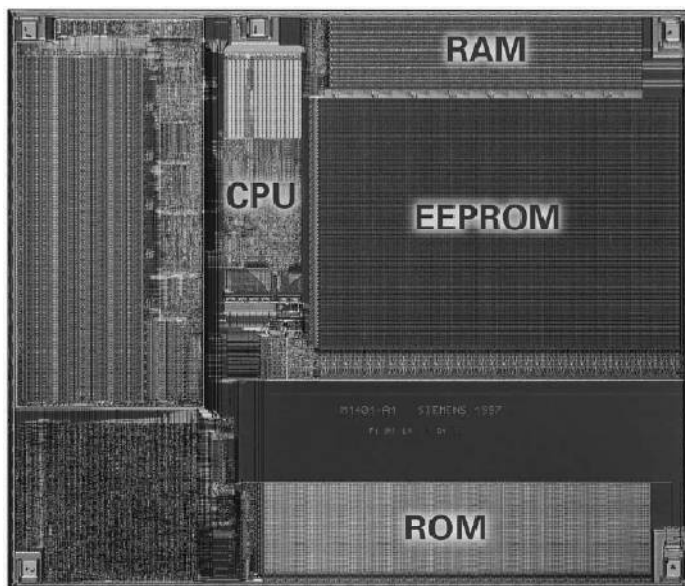
Чипови на смарт картицама који су направљени да би се користили у заштићеним апликацијама, често имају уграђен и криптографски копроцесор. Криптографски копроцесор је специјално интегрисано коло за убрзавање израчунавања специфичних аритметичких и комплексних операција. Без таквих израчунавања је немогуће и замислити коришћење неких од криптографских алгоритама. Картице са криптографским копроцесорима су мало скупље од обичних, али могућности које оне пружају не могу се мерити са могућностима обичних смарт картица [25].

5.3.4. Меморијски систем смарт картице

Поред процесора, једна од најбитних компоненти у смарт картици је и меморија. Различите врсте меморије служе пре свега да чувају програмски код и податке [25].

Смарт картице обично имају три врсте меморије:

- **Стална непроменљива меморија (ROM - Read Only Memory)** се користи за чување фиксних програма на картици. *ROM* меморија се може само читати и у њу се не могу уписивати подаци после њене производње. Процес уписивања бинарне слике (коју представљају програми и подаци) у *ROM* меморију се назива „маскирање“. То се ради у процесу фабричке производње чипа. Није потребно напајање да би се подаци у овој меморији сачували. *ROM* меморија на смарт картици чува рутине оперативног система једнако добро као и перманентне податке и корисничке апликације.
- **Стална променљива меморија (EEPROM – Electrically Erasable Programmable Read-Only Memory)**, као и *ROM* меморија, може да сачува садржај меморије када се искључи напајање. Разлика је у томе што садржај ове врсте меморије може бити мењан у току коришћења картице. Према томе, користи се као складиште података - еквивалент хард диску код *PC* рачунара. Корисничке апликације се такође могу уписати у ову врсту меморије после настанка *EEPROM* меморије. Важна карактеристика *EEPROM* меморије је број циклуса уписивања у меморију у току животног века картице, период задржавања података и време приступа подацима. *EEPROM* у већини смарт картица поуздано може извршити од 100.000 до 1.000.000 уписа и може чувати податке 10 година. Читање података из *EEPROM* меморије је једнако брзо као и читање података из *RAM* меморије, али уписивање у *EEPROM* меморију је око 1.000 пута спорије од уписивања податка у *RAM* меморију.
- **Нестална променљива меморија (RAM - Random Access Memory)** се користи као место где се привремено чувају и обрађују подаци, док картица има напајање. Када картица остане без напајања подаци се бришу. *RAM* меморија која се користи у смарт картицама је статичка *RAM* меморија (*SRAM - Static Random Access Memory*) што значи да се не мора периодично освежавати. *RAM* меморији се може приступати неограничен број пута.



Шема 5-5 Меморијски систем смарт картице

ROM меморија је најјефтинија од све три врсте меморије. *EEPROM* меморијске ћелије заузимају четири пута више места од ћелије *ROM* меморије. *RAM* меморија је веома оскудна у чиповима смарт картица. Ћелије *RAM* меморије заузимају око четири пута више простора од ћелије *EEPROM* меморије.

Данас су друге технологије производње меморије све популарније за коришћење у смарт картицама. *Flash* меморија је врста перзистентне променљиве меморије; прилично је ефикаснија у погледу искоришћења простора и енергије од *EEPROM* меморије. *Flash* меморија се може читати бит по бит, а ажурира се блок по блок. Према томе, *Flash* меморија се обично користи за чување додатних програма или великих података.

5.3.5. Додатни хардвер

Постоје и неки захтеви који се не могу потпуно задовољити коришћењем софтвера, већ само додатним хардвером. Последица је та да различити произвођачи микроконтролера за смарт картице нуде и широк спектар додатних функција у форми хардвера који се налази на чипу.

Једина комуникација између смарт картице и спољашњег света је преко бидирекционог серијског интерфејса. Пренос података преко овог интерфејса је контролисан искључиво од стране оперативног система, без било какве хардверске подршке. То захтева веома комплексан софтвер што представља потенцијалан извор софтверских грешака. Ипак главни проблем је ограничење брзине софтверског преноса података, пошто је и брзина процесора ограничена. Уколико се захтевају веће брзине преноса података неопходно је користити **UART** (*Universal Asynchronous Receiver / Transmitter*). То је компонента опште намене која служи за слање и пријем података независно од процесора. Он није ограничен брзином процесора, нити му је потребан софтвер за комуникацију на нивоу бита. Многи новији типови микроконтролера такође имају и могућност додавања и **USB** (*Universal Serial Bus*) интерфејса као додатка *UART*-у. Са таквим интерфејсом могуће је размењивати податке са терминалом користећи *USB* протокол са хардверском подршком. Нажалост, тренутно није могуће ефективно користити ово *USB* хардверско проширење, све док се *USB* на смарт картицама не регулише стандардом.

Тајмери у микроконтролерима смарт картица су повезани на интерни такт процесора или такт *UART*-а преко делиоца који се може подешавати. Обично имају опсег бројања од 16 или (ређе) 32-bit. Користећи тајмер, број импулса од *start* команде до *end* команде се може мерити без учешћа процесора. Већина тајмера се може користити у моду у којем они одбројавају од унапред дефинисане вредности и испаљују прекид кад бројач досегне нулу, после чега се бројач аутоматски ресетује на иницијалну вредност и наставља да одбројава.

Watchdog тајмер је, такође, често присутан у микроконтролерима. У принципу, он се мора ресетовати после сваке инструкције процесора. Дозвољава процесору да се ресетује у дефинисано стање после дефинисаног максималног интервала уколико буде заробљен у бесконачној петљи.

5.4. Комуникација смарт картица са рачунаром

5.4.1. Уређаји за комуникацију са смарт картицама

За комуникацију са смарт картицама користе се *CAD* уређаји - терминали и читачи картица.



Слика 5-6 Уређаји за комуникацију са смарт картицама

Терминали представљају мини-рачунаре. Они интегришу смарт картице као једну од компоненти. Поред својих функционалности читача смарт картица, терминали могу и да обрађују податке које размењују са смарт картицама. Најчешће се користе у сврхе електронске трговине - у продавницама, на бензинским пумпама, банкоматима и осталим местима где се врши плаћање или подаизање новца помоћу картица.

Читач је повезан на серијски, паралелни или *USB* порт рачунара, преко којег рачунар комуницира са смарт картицом. Читач може остварити комуникацију са картицом контактно или бесконтактно и снабдева картицу енергијом. Иако обични читачи не могу да обрађују податке, већина има функције за детекцију и корекцију грешака у складу са транспортним протоколом.

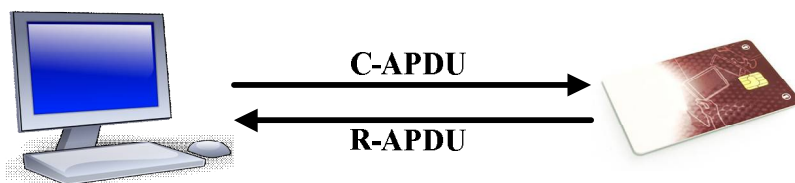
5.4.2. Модел комуникације смарт картица и рачунара

5.4.2.1 *APDU* протокол

Комуникација смарт картица и рачунара, посредством *CAD* уређаја, остварује се разменом података у складу са *APDU* (*Application Protocol Data Units*) протоколом. Као што је специфицирано у *ISO-7816-4* стандарду, *APDU* је протокол на апликационом нивоу смарт картице и апликације домаћина [26].

APDU поруке садржи две структуре:

- **командни *APDU*** (*C-APDU* - *Command Application Protocol Data Units*) - користи је апликација домаћина на *CAD* страни канала за слање порука картици,
- ***APDU* одговора** (*R-APDU* - *Response Application Protocol Data Units*) - користи је картица да пошаље одговоре апликацији домаћина.



Шема 5-7 Модел комуникације смарт картице и рачунара

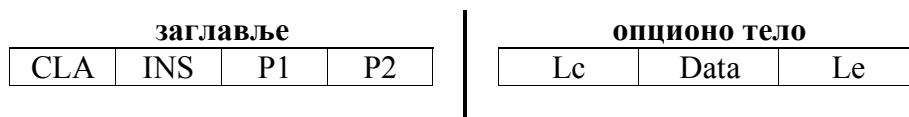
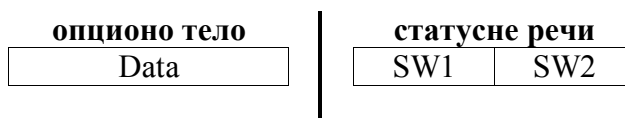
C-APDU и *R-APDU* су увек упарени и функционишу у облику *half-duplex*. Користи се модел *master - slave*, па подаци могу бити послати од домаћина смарт картице или од смарт картице ка домаћину, али не обоје у исто време. Смарт картица увек игра пасивну улогу, чекајући *C-APDU* од рачунара. Она онда извршава инструкцију специфицирану у команди и враћа рачунару *R-APDU*.

Структура *C-APDU* се састоји из следећих елемената:

- **заглавље**
 - *CLA* (*Class of Instruction*) - класа која идентификује категорију *APDU*,
 - *INS* (*Instruction Code*) - код који специфицира инструкцију команде,
 - *P1* и *P2* - параметри за унапређивање квалификације инструкције,
- **опционо тело**
 - *Lc* - специфицира дужину поља са подацима у бајтовима,
 - *Data* - поље са подацима који су потребни картици за извршавање инструкције специфициране у заглављу *APDU*-а,
 - *Le* - специфицира број бајтова који се очекује као одговор.

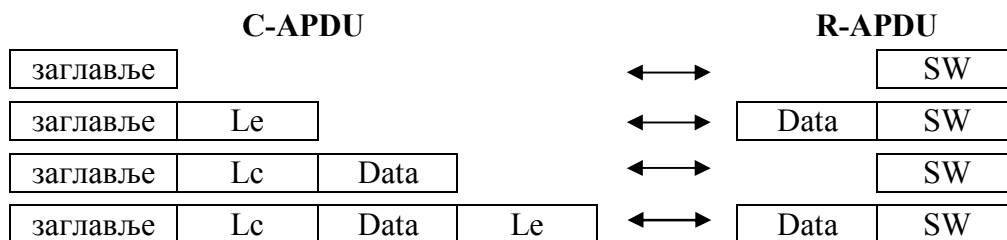
Структура *R-APDU* одговора се састоји из следећих елемената:

- **опционо тело**
 - *Data* - поље са подацима,
- **статусне речи**
 - *SW1* и *SW2* - статусне речи које означавају стање у ком се налази картица после извршења *C-APDU*-а.

Шема 5-8 Структура *C-APDU*Шема 5-9 Структура *R-APDU*

Поље са подацима је опционо и у *C-APDU*-у и у *R-APDU*-у. Према елементима које опциони делови садрже, комуникација *APDU*-а може бити категоризована на следеће случајеве:

1. Нема размене података - *C-APDU* садржи само заглавље, а *R-APDU* само статусну реч.
2. Подаци се не шаљу на картицу, али се са картице враћају неки подаци. Тело *C-APDU* садржи *Le* поље, које специфицира број бајтова податка у *R-APDU*.
3. Подаци се шаљу на картицу заједно са пољем *Le*, али се са картице не враћају подаци.
4. *C-APDU* и *R-APDU* садрже све елементе - подаци се шаљу на картицу и подаци се враћају са картице као резултат извршења команде.



Шема 5-10 Облици командних и одговарајућих APDU-а одговора

5.4.2.2 TPDU протокол

Стандардом *ISO-7816-3* је дефинисан протокол *TPDU (Transmission Protocol Data Units)* који на транспортном нивоу регулише пренос *APDU* порука.

Издавају се две верзије овог протокола у раду са смарт картицама, у зависности од величине података који се преносе и обрађују:

- T=0 протокол који је бајт оријентисан (најмања јединица података је један бајт),
- T=1 протокол који је блок оријентисан (најмања јединица података је блок, а који се састоји од секвенце бајтова).

TPDU структуре које се користе у T=0 протоколу сасвим су различите од оних које се користе у T=1 протоколу.

5.5. Оперативни системи смарт картица

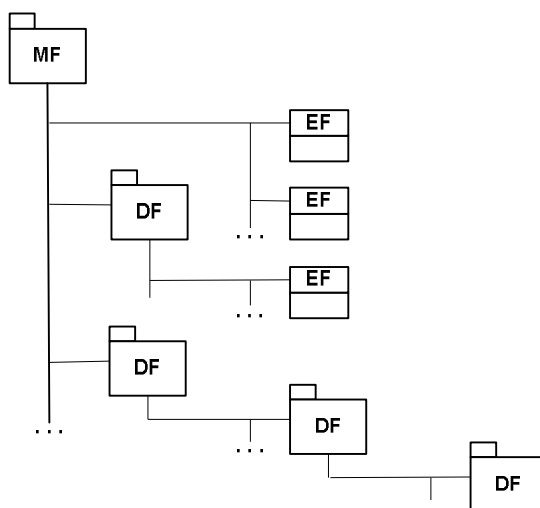
Оперативни системи смарт картица имају сличности са оперативним системима рачунара [27]. Стандард *ISO-7816-4* дефинише одређене инструкције у оквиру *APDU* протокола, које корисничке апликације могу користити. То су најчешће фајл систем оријентисане инструкције, нпр. команде за селекцију и приступ фајловима.

Корисничка апликација је често фајл са подацима специфичним за апликацију, а семантика и инструкције су имплементирани од стране оперативног система. На смарт картицама су углавном заступљени управо оперативни системи оријентисани ка фајл систему. Ипак, све чешће се срећу и новији оперативни системи са слојевитом организацијом и могућношћу учитавања обичног кода апликације.

Смарт картице према *ISO-7816-4* стандарду могу имати хијерархијску архитектуру фајл система. Фајлови су специфицирани идентификатором од 2 бајта и симболичким именом од 16 бајтова. Приступ фајловима је контролисан преко услова приступа, који се могу специфицирати директно за читање фајлова или упис у фајл. Пре сваке операције над фајлом, фајл прво мора бити селектован.

Подржана су три типа фајлова.

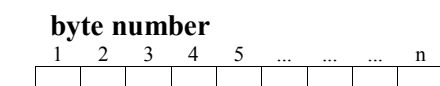
- *Elementary file (EF)* је фајл који садржи податке и не може садржати друге EF.
- *Dedicated file (DF)* представља директоријум на смарт картици који садржи друге DF и EF.
- *Master file (MF)* је корени фајл система. Може да садржи DF и EF. Постоји само један MF на смарт картици и представља специјалан тип DF.



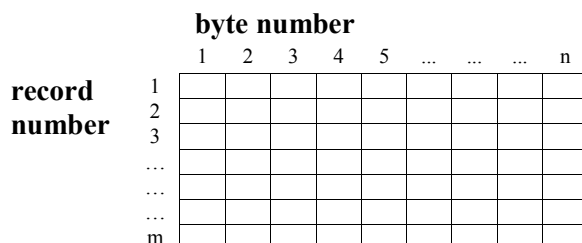
Шема 5-11 Структура фајл система смарт картице

Постоје четири типа елементарних фајлова, у зависности од структуре:

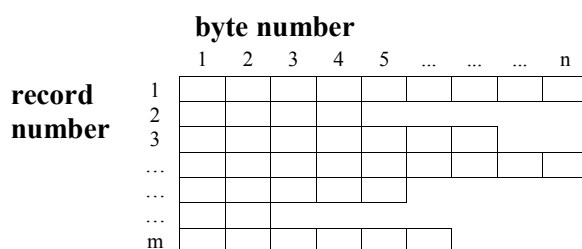
- транспарентни фајл је структуриран као секвенца бајтова са подацима, пошто су остала три *EF* типа структурирани као секвенце појединачних записа (слогова).
- линеарни фиксни фајл има слокове фиксне величине,
- линеарни променљиви фајл има слокове променљиве величине и
- циклични фајл има слокове који су организовани у виду прстена.



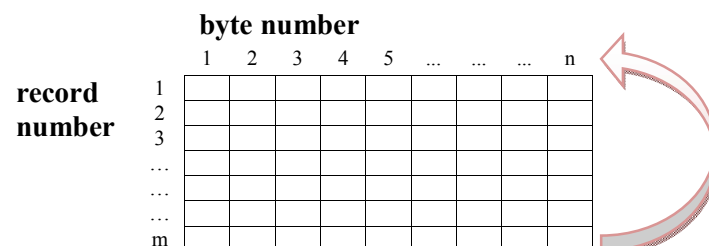
Шема 5-12 Транспарентни фајл



Шема 5-13 Линеарни фајл са фиксном дужином слога



Шема 5-14 Линеарни фајл са променљивом дужином слога



Шема 5-15 Линеарни фајл са цикличном дужином слога

У цикличном фајлу, слогови су у обрнутом редоследу од редоследа у којем се они стављају у фајл - последњи стављени слог се идентификује као слог број 1. Ако се фајл попуни, следећа инструкција за упис ће се преписати преко најстаријег слога у фајлу и постаће слог са редним бројем 1.

5.6. Софтверско окружење

Софтвер који се користи за комуникацију са смарт картицама обично је спрега софтвера картица и софтвера домаћина, тј. рачунара или терминала на који се повезују картице. Савремена окружења за развој и коришћење смарт картица се развијају са главним циљем индустријске производње и интероперабилности различитих софтверско-хардверских елемената који учествују у комуникацији.

Тежишни део софтверског окружења функционише на страни домаћина, који се најчешће састоји из системског софтвера и корисничке апликације:

- Системски софтвер препознаје картицу, обезбеђује подршку за инфраструктуру картице, регулише комуникацију и заштиту, управљање картицом и кључевима итд.
- Корисничка апликација у оквиру система домаћина имплементира инструкције за комуникацију са картицама. Може обезбедити аутентификацију корисника, обраду трансакција и кориснички интерфејс.

Систем картице је софтвер на самој картици. Специфичност софтвера на картици је да се може имплементирати и у асемблеру и у програмском језику. Као код система домаћина, састоји се од системског софтвера и апликације.

- Системски софтвер најчешће контролише рад са меморијом, прихвата комуникацију са домаћином, осигурава интегритет и безбедност података, даје подршку у раду са фајл системом картице и обезбеђује системске сервисе за апликацију на картици.
- Апликација на картици омогућује функције за рад са подацима. С обзиром да системски софтвер најчешће подржава инструкције за рад са подацима, то се оне изостављају код апликација.

5.7. Стандарди

Стандард **ISO-7816** (*Identification cards – Integrated circuit cards with contacts*) дефинише карактеристике контактних чип картица [28]:

- физичке карактеристике,
- димензије и положај контаката,
- електричне сигнале и транспортне протоколе,
- међуиндустријске наредбе,
- идентификатор апликације,
- међуиндустријске елементе података,
- међуиндустријске *SCQL* наредбе.

GSM стандард је издао *ETSI* (*European Telecommunications Standards Institute*) и покрива коришћење смарт картица у јавним и мобилним телефонским системима. *GSM*

стандарди су спецификација међународног земаљског мобилног телефонског система [29]. Значајни су :

- *GSM 11.11* – спецификација *SIM* модула,
- *GSM 11.14* – спецификација алата за развој апликација *SIM* модула,
- *GSM 03.48* – сигурносни механизми алата за развој апликација *SIM* модула,
- *GSM 03.19* – *SIM API* за *Java Card* платформу.

EMV (*Europay, MasterCard and Visa*) спецификацију су дефинисали *Europay, MasterCard* и *Visa*. Она је настала као проширење *ISO-7816* стандарда у смеру специфичних потреба финансијске индустрије [30]. Састоји се од четири дела:

- *Book 1* - спецификација захтева за интерфејс између картице и терминала,
- *Book 2* - спецификација механизма *EMV* заштите и процес управљања кључевима,
- *Book 3* - спецификација апликација,
- *Book 4* - спецификација захтева за интерфејс према платним системима.

Open Platform стандард дефинише интегрисано окружење за развој и употребу вишенамених система картица. *Open Platform* стандард садржи спецификацију картице и спецификацију терминала. Спецификација картице дефинише комуникацију картице са терминалом и *management* апликације на картици. Спецификација терминала дефинише архитектуру апликације терминала. Терминал мора задовољити *ISO* и *EMV* стандарде. *Open Platform* спецификације у почетку је развијала *Visa*. Данас су оне део *GlobalPlatform* организације [31].

OpenCard Framework, иницијално произашао из *IBM*-а (*International Business Machines*), власништво је *OpenCard* конзорцијума. *OpenCard* конзорцијум обухвата све важније учеснике индустрије смарт картица [32]. *OpenCard Framework* дефинише стандардно окружење за писање апликација терминала и окружење за комуникацију са читачима картица и апликацијама на смарт картицама. Архитектура *OpenCard Framework* модела раздваја функционалност система са смарт картицама између произвођача терминала за картице, произвођача оперативних система за смарт картице и издавача картица. Циљ је постићи међусобну независност наведених страна. Имплементиран је у програмском језику *Java*. На *Windows* платформама *OpenCard Framework* приступа *CAD* уређају кроз инсталирани *PC/SC (Personal Computer/Smart Card) manager*.

PC/SC спецификација власништво је *PC/SC WorkGroup*, индустријског конзорцијума који обухвата све важније учеснике у индустрији смарт картица. *PC/SC* дефинише архитектуру намењену коришћењу смарт картица у домену персоналних рачунара. У тој архитектури апликација терминала се гради над *PC/SC* сервисом и *PC/SC manager*-ом. Сервиси енкапсулирају функционалности и специфичности појединих смарт картица и омогућују да се картици приступи кроз програмерско решење високог нивоа, а *PC/SC manager* повезује оперативни систем персоналног рачунара са *CAD* уређајем [33].

6. Java окружење

6.1. Java платформа

Java програмски језик има доста сличности у спецификацијама са другим објектно оријентисаним програмским језицима. Код написан у програмском језику *Java* се преводи за извршавање на *Java* платформи. Секвенца *bytecode* асемблерских инструкција *Java* процесора се компајлира у секвенцу инструкција типичних за платформу на којој се извршава код [34].

Just-In-Time (JIT) компајлери реализују превођење *Java bytecode* инструкција за извршење на непосредној платформи (*native code*). Ови компајлери у великој мери убрзавају извршавање кода [35].

На овај начин се остварује модел портабилних апликација на нивоу извршног кода, који због својих позитивних карактеристика има немерљиву улогу у областима употребе смарт картица, микрорачунара и мобилних уређаја. Омогућен је развој софтвера за различите уређаје са подршком за *Java*-у уз ослањање на стандардне асемблерске инструкције за различите производе и употребу [36].

Java Virtual Machine (JVM) представља платформу за извршавање програма у чијој основи се налази програмски модел имагинарног *Java* процесора [37].

6.2. Java Cryptography Architecture

Java Cryptography Architecture (JCA) као имплементирана криптографска технологија у оквиру *JDK Security API (Java Development Kit Security Application Programming Interface)*, представља транспарентан модел за развој и пружање разноврсних криптографских механизма на *Java* платформи [34].

JCA раздваја криптографске поступке и методе од алгоритамске имплементације и на тај начин омогућује реализацију различитих криптографских алгоритама и функција [38].

Основу криптографских подсистема чине класе пакета ***java.security: provider*** и ***security***.

Provider - представља асоцијативни низ у коме је кључ - назив алгоритма, а вредност - пуно име класе која реализује дати алгоритам. Изведена је из класе ***Properties*** пакета *java.util*.

Security класа садржи листу провајдера и методе карактеристичне за листе. Да би се користиле имплементације алгоритама које пружа одређени провајдер, неопходно је да се дати провајдер региструје у листу објекта *Security*.

6.3. Cryptographic Service Provider

Cryptographic Service Provider (CSP), тј. провајдер криптографских услуга, омогућује имплементацију следећих механизма [39]:

- база за архиву кључева (*keystore*),
- алгоритам дигиталног потписа (*DSA*),
- алгоритам за генерисање отисака порука *MD5* и *SHA-1*,
- генерисање и управљање параметрима алгоритма,
- подршка за конверзију између различитих типова кључева,
- подршка за сертификате према стандарду *X.509*,
- подршка за генерисање листа за опозив сертификата (*CRL*),
- генератор псеудо-случајних бројева итд.

6.4. Security Policy

Security Policy, тј. сигурносно правило, представља политику коришћења ресурса система. Да би одређени код имао приступ неком ресурсу система, потребно је да је то дефинисано одговарајућом дозволом у сигурносној политици система. Дозвола садржи назив (*Target Name*) и листу дозвољених или недозвољених акција [40].

6.5. Security Manager

Security Manager служи за проверу сигурносне политике имплементираних на систему. Регулације коришћење ресурса система удаљеном коду за који нису дефинисане дозволе сигурносном политиком. Уколико се *Java* код покреће локално, сигурносни менаџер се не користи аутоматски, већ апликација има потпун приступ ресурсима [40].

6.6. Java Secure Socket Extension

Java Secure Socket Extension (JSSE) представља надоградњу сигурносних механизма. *JSSE* је доступан за директну интеграцију у апликације и пружа апстракцију сигурносних алгоритама и механизма успоставе комуникације (*handshaking mechanisms*). Тиме се умањују безбедносни ризици. *JSSE* пружа *API* према сигурним протоколима *SSL* и *TLS*, као и према системима других провајдера криптографских услуга, најчешће комерцијалних [41].

6.7. Java Cryptography Extension

Java Cryptography Extension (JCE) је још једно проширење *JDK*, за потребе енкрипције, размену кључева и аутентификацију порука. *JCE* је у спрези са криптографским аспектима *JDK* и представља комплетан криптографски *API*

независтан од платформи. Тиме је омогућено додавање нових сигурносних провајдера услуга од компанија из области сигурносне технологије [42].

Класе овог *API*-а се користе у алгоритмима са циљевима:

- криптографских операција (дигитални потпис, отисак поруке),
- генерисања криптографских кључева или параметара и
- креирања сигурносних криптографских објеката (базе кључева или сертификата).

Примери класа су:

- *Signature* - пружа приступ функционалности различитих криптографских алгоритама дигиталног потписа и
- *KeyFactory* - пружа пар кључева јавни / тајни у формату који методе *initSign* и *initVerify* могу користити над *Signature* објектом.

6.8. Управљање кључевима и сертификатима

Keystore је база кључева и сертификата, чији називи представљају *alias*-е [38].

Keytool је алат за управљање базом кључева [43]. Има разноврсне могућности употребе и може служити за:

- креирање парова јавни / тајни кључ,
- издавање захтева за сертификатима који се шаљу одговарајућем *CA*,
- увођење добијених одговора, односно сертификата и
- одређивање јавних кључева којима се верује и управљање заштићеним базама кључева и сертификата.

Truststore је посебна база кључева за чување јавних кључева сертификационих тела за верификацију и других сертификационих ауторитета. У *Java* постоји предефинисана база *truststore* у датотеци *cacert*.

Наредба	Функција
- <i>certreq</i>	генерише захтев за потврдом сертификата (<i>CSR</i>) који се шаље телу задуженом за издавање сертификата (<i>CA</i>) како би потврдили ваљаност јавног кључа
- <i>delete</i>	брише одређени унос у бази кључева
- <i>export</i>	експортује сертификат јавног кључа за одређени ентитет у датотеку
- <i>genkey</i>	креира унос у бази кључева са генерисаним тајним и одговарајућим јавним кључем који се ставља у самопотписани сертификат
- <i>import</i>	увози сертификат који се третира као поуздан сертификат или као одговор тела за издавање сертификата (<i>CA</i>) који ће заменити самопотписани сертификат у бази сертификата
- <i>keypasswd</i>	додељује шифру тајном кључу
- <i>list</i>	исписује уносе у бази кључева
- <i>printcert</i>	исписује информацију у датотеци која садржи сертификат
- <i>storepasswd</i>	додељује шифру читавој бази кључева

Табела 6-1 Наредбе *keytool* алата

Наредба	Одређује
- <i>storetype</i> <i>storetype</i>	тип базе кључева који се инстанцира
- <i>keystore</i> <i>keystore</i>	назив и опционо путању до базе кључева
- <i>storepass</i> <i>storepass</i>	шифру којом се штити комплетни <i>keystore</i>
- <i>v storepass</i>	детаљни испис информација о сертификату
- <i>alias</i> <i>aliasName</i>	нови назив који се користи за одређени унос у базу кључева
- <i>validity</i> <i>numDays</i>	дужину трајања одређеног сертификата (подразумевано је 90 дана)
- <i>keypass</i> <i>keypass</i>	шифру која се користи за заштиту тајног кључа појединог уноса у бази кључева

Табела 6-2 Опције наредби *keytool* алата

7. Java Card технологија

Java Card технологија омогућује програмима који су писани у програмском језику *Java* да буду извршени на смарт картицама и другим уређајима са ограниченим ресурсима [44].

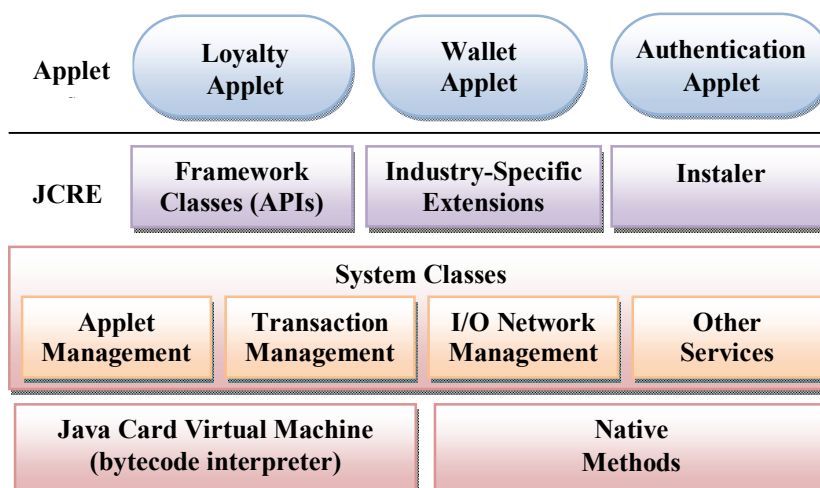
С обзиром да смарт картице представљају једну од најмањих рачунарских платформи која се данас користи, најчешће капацитета меморија реда *RAM 1KB*, *EEPROM 16KB* и *ROM 24KB*, највећи изазов у *Java Card* технологији је обезбеђење довољног простора за апликације које би се извршавале на самим картицама. Резултат тога је подршка само делу карактеристика *Java* програмског језика и подела *Java* виртуелне машине на део који се налази изван смарт картице и део на самој картици. Многи задаци обраде који нису ограничени да се извршавају у *run-time*-у, као што су динамичко учитавање класа, верификација *bytecode*, линковање и оптимизација, поверене су виртуелној машини која се налази изван смарт картице, где ресурси не представљају проблем.

Осим што обезбеђују подршку за програмирање у *Java*, *Java Card* технологија дефинише и извршно окружење које подржава меморију смарт картице, комуникације, заштиту и моделе извршавања апликација. *Java Card* извршно окружење прилагођено је интернационалном стандарду за смарт картице *ISO-7816*.

Најзначајнија предност *Java Card* извршног окружења је то што обезбеђује чисту поделу између смарт картице и апликација. Извршно окружење енкапсулира сложеност и детаље који леже испод система са смарт картицама, а апликације захтевају системске сервисе и ресурсе кроз добро дефинисани интерфејс за програмирање високог нивоа.

7.1. Java Card архитектура

Java Card је платформа на којој се апликације писане у *Java* програмском језику могу покренути на смарт картицама и другим меморијски ограниченим уређајима [45]. Из разлога подељене архитектуре виртуелне машине, ова платформа је дистрибуирана између смарт картице и окружења на рачунару.



Шема 7-1 Архитектура Java Card технологије

Архитектура *Java Card* технологије се састоји из три дела, од којих је сваки дефинисан посебном спецификацијом:

- ***The Java Card Application Programming Interface (API)*** - описује скуп основних и додатних *Java* пакета и класа за програмирање апликација за смарт картице.
- ***The Java Card Runtime Environment (JCRE)*** - описује понашање *Java Card* извршног окружења, укључујући рад са меморијом, рад са аплетима и друге извршне особине.
- ***The Java Card Virtual Machine (JCVM)*** - дефинише подскуп *Java* програмског језика и дефиницију виртуелне машине подесне за апликације на смарт картици.

7.2. Java Card Runtime Environment

Java Card Runtime Environment (JCRE) је извршно окружење смарт картице. Налази на врху хардвера смарт картице и састоји се од системских компоненти *Java Card*-а, тј. *Java Card* виртуелне машине (интерпретер *bytecode*), *Java Card API* класа, специфичних индустријских проширења и *JCRE* системских класа [46].

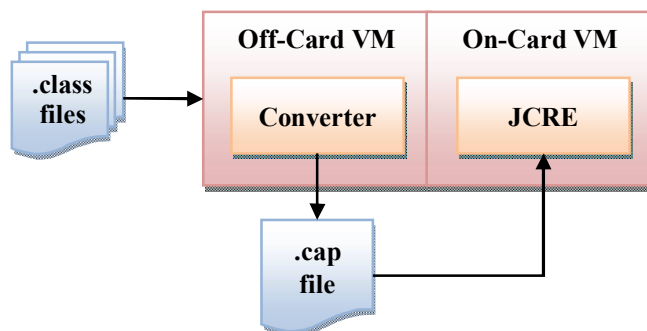
JCRE је одговорно за управљање ресурсима на картици, комуникацију са другим уређајима, извршавање аплета, сигурност система и аплета који се налазе на картици. Према томе, он има улогу оперативног система.

JCRE добро раздваја аплете од приватних технологија које нуде произвођачи смарт картица и обезбеђује стандардне системске и *API* интерфејсе за аплете. Аплети се лакше пишу и преносиви су на различите архитектуре смарт картица.

7.3. Java Card Virtual Machine

Java Card Virtual Machine (JCVM) извршава *bytecode*, контролише алокацију меморије, управља објектима и подстиче сигурност при извршавању. Уграђене методе обезбеђују подршку *JCVM*-у и системским класама вишег слоја. Оне су одговорне за прихватање комуникационих протокола ниског нивоа, управљање меморијом, подршку криптографији итд [47].

Основна разлика између *JCVM* и *JVM* је то што је *JCVM* имплементирана као два одвојена дела. Део који се налази на смарт картици садржи *Java Card bytecode interpreter*, док се на рачунару или радној станици налази *Java Card konverter*. Заједно имплементирају све функције виртуелне машине – учитавање и извршавање *Java* класа.



Шема 7-2 Структура Java Card Virtual Machine

7.4. Формати фајлова

Java Card технологија користи два формата бинарних фајлова који омогућавају развој, дистрибуцију и извршавање *Java Card* софтвера независно од платформе.

- *Cap* фајл садржи извршну бинарну репрезентацију класа у *Java* пакету.
- *Jar* формат фајлова се користи као контејнер за *Cap* фајлове.

Cap фајл је *Jar* фајл који садржи скуп компоненти, од којих је свака чувана као посебан фајл у *Jar* фајлу. Свака компонента описује један аспект садржаја *Cap* фајла, као што су информације о класама, извршним *bytecode*, информацијама за линковање, информацијама за верификацију, итд. Формат *Cap* фајла је оптимизован. Он дефинише *bytecode* инструкцијски скуп који је заснован и оптимизован из *Java bytecode* инструкцијског скупа.

Class фајл је централни део *Java* архитектуре. Он дефинише стандард за бинарну компатибилност *Java* платформе. Због карактеристика *Java Card* архитектуре система, *Cap* фајл је постао стандард за бинарну компатибилност *Java Card* платформе. *Cap* фајл је формат у којем се софтвер учитава на смарт картице. На пример, *Cap* фајлови дозвољавају динамичко учитавање аплета након што се картица произведе.

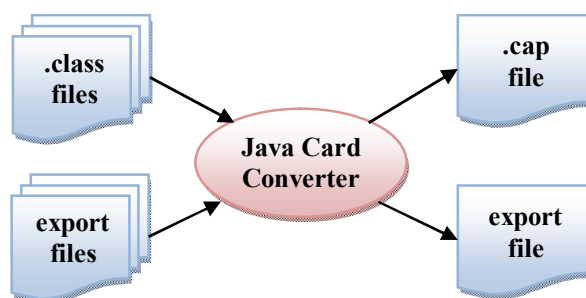
Export фајлови се не учитавају на смарт картицу и према томе се не користе директно на интерпретеру. Њих прави и користи конвертер за верификацију и линковање. *Export* фајл садржи јавне *API* информације о унутрашњости пакета класа. Он дефинише обим приступа и имена класа и обим приступа и потписе метода и поља класе. *Export* фајл такође садржи информације за линковање које користе решавање референце између пакета који су на смарт картици. *Export* фајлови не садрже никакву имплементацију, они не садрже бајткод. То значи да се *Export* фајл може слободно дистрибуирати од стране програмера аплета ка потенцијалним корисницима аплета.

7.5. Java Card Converter

Java Card Converter се налази на делу виртуелне машине који је изван смарт картице. Насупрот *Java* виртуелној машини, која обрађује једну класу у једном тренутку, јединица коју конвертује конвертер је пакет. *Class* фајлове прави *Java* компајлер из изворног кода. Тада, конвертер обрађује све *Class* фајлове који граде *Java* пакет и конвертује пакет у *Cap* фајл.

Током конверзије, конвертер извршава задатке које *Java* виртуелна машина на *desktop* окружењу ради у време учитавања класа:

- верификује да ли су учитане бинарне слике *Java* класа добро формиране,
- проверава да ли је *Java Card* подскуп инструкција нарушен,
- врши иницијализацију статичких променљивих,
- разбија симболичке референце класа, метода и поља у компактније форме тако да се могу користити много ефикасније на картици,
- оптимизује *bytecode* користећи информације које је добио у току учитавања класа и линковања и
- алоцира складишта података и креира структуре података које виртуелна машина користи за репрезентацију класа.



Шема 7-3 Конвертовање пакета

Конвертер као улаз користи *Class* фајлове које треба да конвертује и један или више *Export* фајлова. Осим прављења *Cap* фајла, конвертер генерише и *Export* фајл за конвертовани пакет. Конвертер учитава све класе из *Java* пакета. Уколико пакет увози класе из других пакета, конвертер такође учитава и *Export* фајлове тих пакета. Излаз конвертера је *Cap* фајл и *Export* фајл за пакет који је конвертован.

Како *Java Card* технологија подржава само део *Java* програмског језика, а *JCVM* само карактеристике које захтева подскуп програмског језика, сваку карактеристику језика која није подржана, а искоришћена је у аплету конвертор ће детектовати.

7.6. Java Card Interpreter

Java Card Interpreter обезбеђује извршну подршку моделу *Java* програмског језика и према томе омогућује независност кода аплета од хардвера. Његова улога је да:

- извршава *bytecode* инструкције и начелно извршава аплете,
- контролише алокацију меморије и креирање објеката,
- игра пресудну улогу у обезбеђивању сигурности у извршавању.

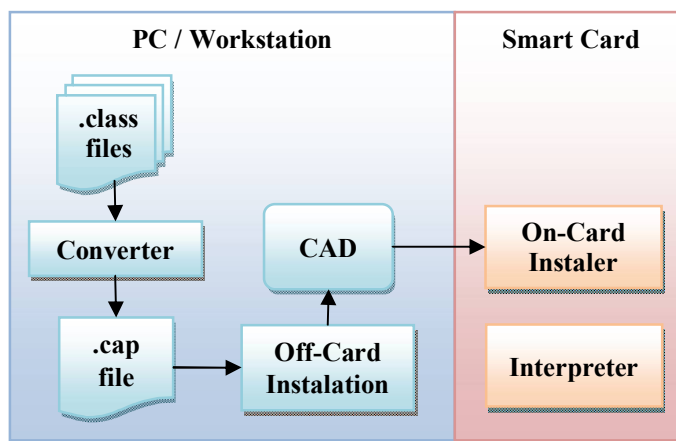
7.7. Java Card Instaler и Off-Card инсталациони програм

Java Card интерпретер не учитава сам *Cap* фајлове. Он једино извршава код који пронађе у *Cap* фајлу. У *Java Card* технологији, механизми за учитавање и инсталирање *Cap* фајла су уграђени у јединицу која се назива *Java Card Instaler*.

Java Card Instaler омогућава заштићено учитавање софтвера и аплета на картицу после производње картице и њеног издавања власнику картице. Инсталер је опциона компонента *JCRE*-а. Без инсталера, сав софтвер за картицу, укључујући аплете, мора бити уписан у меморију картице током производње картице.

Java Card Instaler се налази на картици. Он ради заједно са *off-card* инсталационим програмом. *Off-card* инсталациони програм преноси бинарни код *Cap* фајла до инсталера преко *CAD* уређаја. Инсталер уписује бинарни код у меморију смарт картице, повезује га са осталим класама које се налазе на картици, а затим креира и иницијализује било коју структуру података која се користи интерно од стране *Java Card* извршног окружења.

Поделом функционалности, инсталер обезбеђује да интерпретер буде мали и обезбеђује флексибилност имплементације инсталера.



Шема 7-4 Java Card Intaler и Off-Card инсталациони програм

7.8. Системске класе

Системске класе се понашају као да се извршавају на *JCRE*-у. Оне су аналогне језгру оперативног система. Системске класе су допуна управљању трансакцијама, управљању комуникацијама између апликације домаћина и *Java Card* аплета и контролисању креирања, селектовања и деселектовања аплета.

7.9. Java Card Applets

Java Card Applets се не смеју помешати са *Java* аpletима само због свог назива. *Java Card Applets* су *Java* програм који се држе скупа конвенција које им дозвољавају да се извршавају у извршном окружењу *Java Card*-а. *Java Card Applets* нису намењени да се покрећу унутар окружења претраживача. Разлог избора имена аplet за *Java Card* апликације је тај што се *Java* аплети могу учитати у *Java Card* извршно окружење после производње картице. То значи, за разлику од апликација у многим уграђеним системима, аплети не морају бити уписани у *ROM* током производње, већ се могу касније динамички учитати на картицу.

Класа аплета мора бити наслеђена из класе *javacard.framework.applet*. Основна класа *Applet* је суперкласа за све аплете који се налазе на *Java* смарт картици. *Applet* класа представља шематски отисак који дефинише променљиве и методе у аpletу. *Applet* који се извршава на картици је инстанца класе *Applet*. Као и сваки перзистентни објект једном креиран, *Applet* егзистира на картици заувек.

Java Card извршно окружење подржава окружење у којем се може извршавати више апликација. Више аплета може коегзистирати на једној *Java* смарт картици, и аplet може имати више инстанци.

7.10. Пакети и конвенције именовања аплета

Пакети и програми који се користе у *Java* платформи су јединствено идентификовани са *Unicode* стринговима и шемом именовања базираној на именима интернет домена. На *Java Card* платформи, свака инстанца аплета се јединствено именује и селекује од стране апликације на основу идентификатора (*AID* – *Applet Identifier*). Такође, *Java* пакету се додељује *AID*. Када се учита на картицу, пакет се повезује са другим пакетима на картици преко њихових *AID*-а.

ISO-7816 означава *AID*-е као јединствену идентификацију апликације која је на картици и заштићених врста фајлова на системима са картицама. *AID* је низ бајтова који се могу интерпретирати у два одвојена дела.

RID (5 bytes)	PIX (0 – 11 bytes)
---------------	--------------------

Шема 7-5 Applet Identifier

Први део је вредност од 5 бајта која се назива *RID* (*Resource Identifier*). Други део је вредност променљиве дужине која се назива *PIX* (*Proprietary Identifier Extension*). *PIX* може бити величине од 0 до 11 бајта. *AID* може имати од 5 до 16 бајта величине.

ISO контролише доделу *RID*-ова компанијама: свака компанија има јединствени *RID*. Компаније управљају додељивањем *PIX*-ом у *AID*-има.

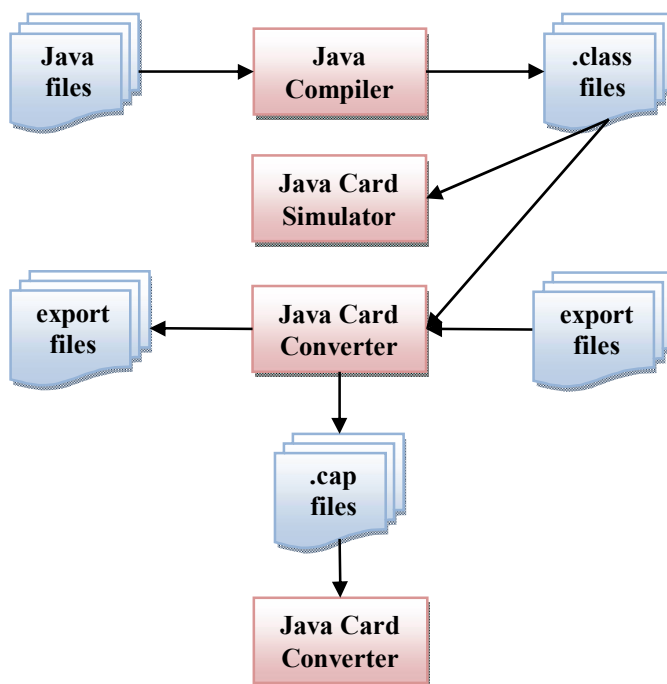
На *Java Card* платформи, *AID* за сваки пакет се добија тако што се надовезују компанијски *RID* и *PIX* за тај пакет. *AID* аплета се прави на сличан начин као и *AID* пакета. Он се саставља од *RID*-а оног ко је направио аплет и *PIX*-а за тај аплет. *AID* аплета не сме да има исту вредност као и неки други *AID* пакета или другог аплета. Ипак, пошто *RID* у *AID*-у идентификује произвођача аплета, *AID* пакета и *AID*-и аплета дефинисаних у истом пакету морају да деле исте *RID*.

AID пакета и *AID* аплета за сваки аплет који је дефинисан у пакету је специфициран у *CAP* фајлу. Они се додељују у конвертеру када се генерише *CAP* фајл.

7.11. Процес развоја аплета

Развој *Java Card* аплета почиње као и било који други *Java* програм: програмер пише једну или више *Java* класа и компајлира изворни код *Java* компајлером, правећи, при томе, један или више *Class* фајлова [48].

Аплет се извршава, тестира и отклањају се грешке у симулираном окружењу. Симулатор симулира *Java Card* извршно окружење на рачунару или радној станици. У симулираном окружењу, аплет се извршава на *JVM*, и тада се *Class* фајлови аплета извршавају. У том случају симулатор може искористити многе развојне алате *Java* (виртуелну машину, *debugger* итд.) и дозволити програмеру да тестира понашање аплета и брзо види резултате аплета без проласка кроз процес конверзије. Током овог корака се тестирају глобални функционални аспекти аплета. Ипак, не могу се тестирати све особине *Java Card* виртуелне машине као што су заштитна баријера и понашање транзистентних и перзистентних објеката.



Шема 7-6 Процес развоја аплета

Тада се *class* фајлови аплета који чине *Java* пакет конвертују у *Cap* фајл користећи *Java Card* конвертер. *Java Card* конвертер као улаз узима не само *class* фајлове који треба да се конвертују, него и један или више *Export* фајлова. Када се конвертује пакет у којем је и аplet, конвертер такође може да направи и *Export* фајл за тај пакет. *Cap* фајл или *export* фајл представљају *Java* пакет. Уколико аplet укључује неколико пакета, *CAP* фајл и *export* фајл се креирају за сваки пакет.

У следећем кораку, *Cap* фајл(ови) који представља аplet се учитава и тестира у емулираном окружењу. Емулатор такође симулира *Java Card* извршно окружење на рачунару или радној станици. Ипак, емулатор је веома софистицирано средство за тестирање. Он обухвата имплементацију *JCVM*. Понашање аплета током извршавања на емулатору треба да буде исто као и понашање на правој смарт картици. У овој фази развоја не тестира се само развој аплета него се мери и његово понашање у току извршавања.

Већина *Java Card* симулатора и емулатора долазе заједно са *debugger*-ом. *Debugger* дозвољава програмеру да поставља тачке прекида програма, посматрајући стање извршавања аплета у симулираном или емулираном *Java Card* извршном окружењу. На крају, када је аplet тестиран, представљен као један или више *Cap* фајлова, учитава се и инсталитра на *Java* смарт картицу.

8. Реализација апликације

8.1. Идеја решења

Део технологија описаних у раду су примењене у реализацији апликације која омогућава креирање дигиталног сертификата као профила корисника према задатим параметрима и његово смештање на смарт картицу, проверу права приступа и електронско потписивање фајлова, са циљем обезбеђивања аутоматизације процеса контроле права приступа, праћења активности корисника и непорецивости трансакција и интегритета података у оквиру информационог система.

8.2. Архитектура апликације

Програмско решење се заснива на примени програмског језика *Java*, *Java Card* технологије и инфраструктури јавног кључа.

Апликација поседује класичну вишеслојну архитектуру из разлога обезбеђења једноставности имплементације могућих надоградњи у функционалности апликације.



Шема 8-1 Архитектура вишеслојне апликације

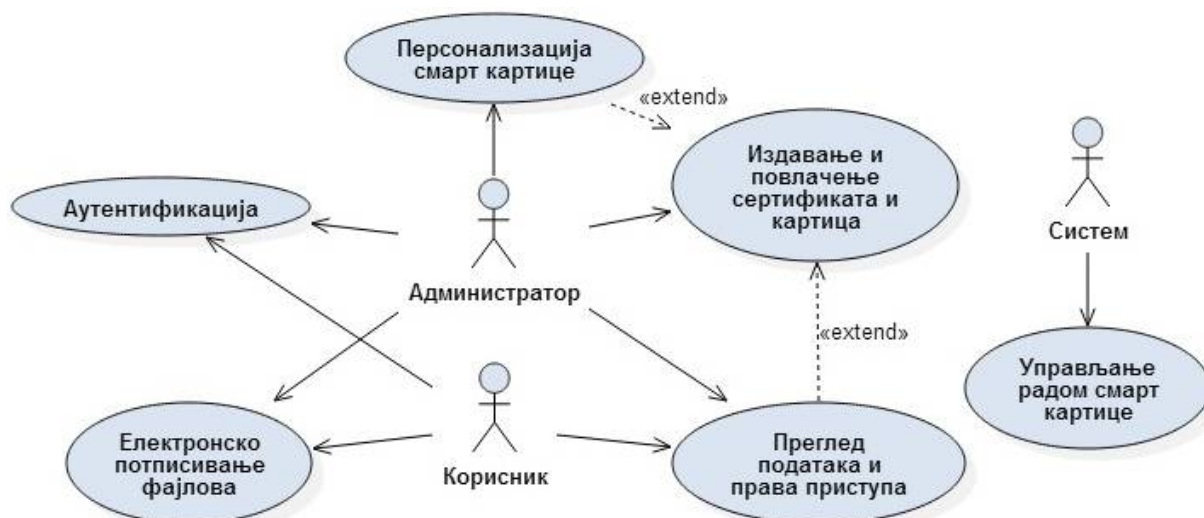
У најнижем слоју се налазе **смарт картице**. Између **корисника** и смарт картица постоје **логика система** и **кориснички интерфејс**, који функционишу на рачунарима и терминалима на које се повезују смарт картице.

Логика система има улогу посредника између корисника и смарт картице, обезбеђује поузданост функционисања система и није видљива кориснику система.

8.3. Овлашћена лица

У оквиру система постоји **администратор**, тј. инстанца која издаје и повлачи легитимације за кориснике. Он генерише сертификат и на картицу уписује личне податке корисника и врши исправке евентуалних грешака у раду са картицама. Дозвољено му је и преглед података на картици и електронско потписивање фајлова.

Корисницима је омогућен увид у садржај података на картици и дозвољених права приступа, као и електронско потписивање докумената.



Дијаграм 8-2 UML дијаграм основних функционалности система у интеракцији са корисницима

Апликација треба да обезбеди да се одабране функционалности које обезбеђује смарт картица могу једноставно искористити. Управљање радом смарт картице означава сталну контролу над радом картице у виду провере да ли је картица убачена у читач и да ли картица извршава задате инструкције. Поред тога, апликација треба да обезбеди аутентификацију администратора и корисника [49].

8.4. Логички модел и имплементација

Дефинисано је да се смарт картица издаје у виду јединствене персоналне легитимације која садржи сва поља у складу са стандардом *X.509*. Поред профила корисника и личних података, садржи податке који ће корисника једнозначно одређивати, као и податке који ће служити за ауторизацију корисника, односно који ће одређивати његова права приступа.

С обзиром да се подаци на картици чувају у облику дигиталног сертификата, потребно је да у ту сврху садрже и одговарајуће податке:

- серијски број,
- податке о идентитету корисника коме је издат сертификат (кориснички *ID*, име и презиме, датум рођења итд.),
- *PIN* код,
- алгоритам дигиталног сертификата,
- дигитални сертификат,
- улога / ниво корисника,
- јавни криптографски кључ корисника сертификата,
- податке о сертификационом телу које је издало сертификат,
- период важења (датум креирања и датум истека важења).

Серијски број се користи за потребе ауторизације и провере на *CRL*.

Кориснички ID је број који једнозначно одређује корисника. Уколико се кориснику у неком будућем периоду поништи стара картица, а изда нова, остаће му исти кориснички *ID*. На тај начин ће систем у ком је интегрисана апликација за рад са картицама недвосмислено знати о ком кориснику је реч. Природно решење за кориснички *ID* је ЈМБГ, као јединствена карактеристика лица.

Постојање ***PIN* кода** елиминише могућност злоупотребе картице у случају крађе или губитка. Идентификација помоћу *PIN* кода безбеднија је од других начина идентификовања из разлога јер не путује мрежом, отпорна је на нападе типа *brutforce* или *dictionary*, полисе које регулишу дужину и учестаност промене *PIN*-а могу бити мање рестриктивне од оних за *password* итд. [6].

Улога / ниво корисника директно одређује која ће права приступа имати корисник, тј. којим ће сервисима и подацима моћи да приступи у оквиру система.

Легитимација и сертификат би требали бити важећи за одређен **временски период** и због тога је потребно да постоји податак о датумима почетка и истека важења.

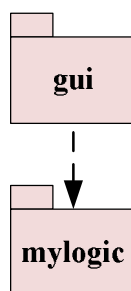
Комуникација са читачем за смарт картице остварује се преко *Java Smart Card I/O API* [50].

Решењем је предвиђено да организација у оквиру које функционише систем поседује своје сертификационо тело, које је одговорно за креирање, издавање и повлачење дигиталних сертификата и легитимација.

Као и свако сертификационо тело, њега карактеришу следећи елементи:

- сопствени приватни криптографски кључ,
- сопствени дигитални сертификат са одговарајућим јавним криптографским кључем и са његовом путањом,
- листа опозваних сертификата.

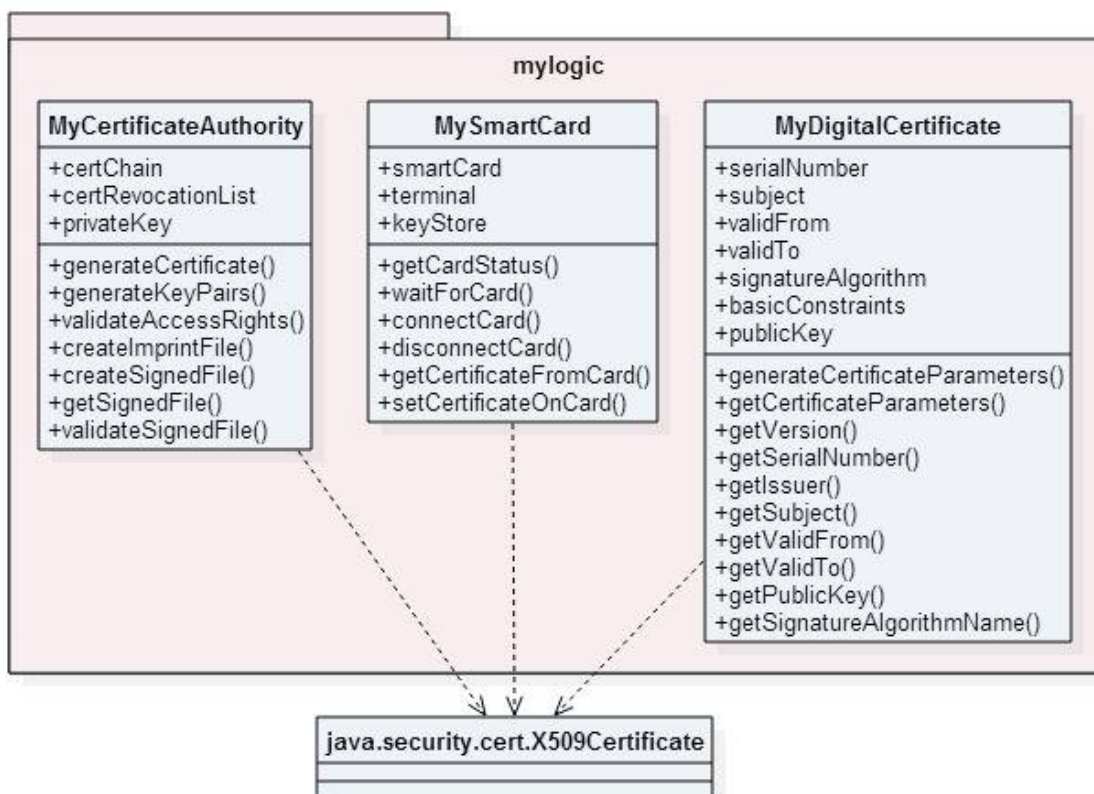
Логички модел апликације се састоји из основних пакета: *gui* и *mylogic*.



Дијаграм 8-3 UML дијаграм основних пакета

Пакет ***gui*** садржи класе које имплементирају графички кориснички интерфејс (*GUI* - *Graphical User Interface*) за одговарајуће случајеве употребе предвиђене решењем, као што су издавање легитимација (*GenerateCertificateGUI*), унос *PIN* кода (*EnterPinGUI*), проверу права приступа (*ValidateAccessRightsGUI*) итд. *GUI* је реализован употребом пакета *javax.swing* и *java.awt* [51].

Класе пакета ***mylogic*** представљају апстракције дигиталног сертификата који се креира као профил корисника и који се смешта на смарт картицу. У вези зависности су са класом *java.security.cert.X509Certificate*.



Дијаграм 8-4 UML дијаграм класа у пакету mylogic

Апстракција сертификационог тела је имплементирана класом **MyCertificateAuthority**. Она између осталог садржи следеће методе:

- *generateCertificate()* - креирање дигиталног сертификата,
- *generateKeyPairs()* - генерисање пара кључева,
- *validateAccessRights()* - провера права приступа,
- *createSignedFile()* - дигитално потписивање фајлова и
- *validateSignedFile()* - валидација дигитално потписаних фајлова.

Поред наведених метода, садржи и варијабле:

- *privateKey* - приватни кључ који се користи за потписивање сертификата,
- *certificateChain* - сертификациони ланац у који су укључени сопствени сертификат сертификационог тела и остали сертификати у хијерархији,
- *CRL* - листа повучених сертификата.

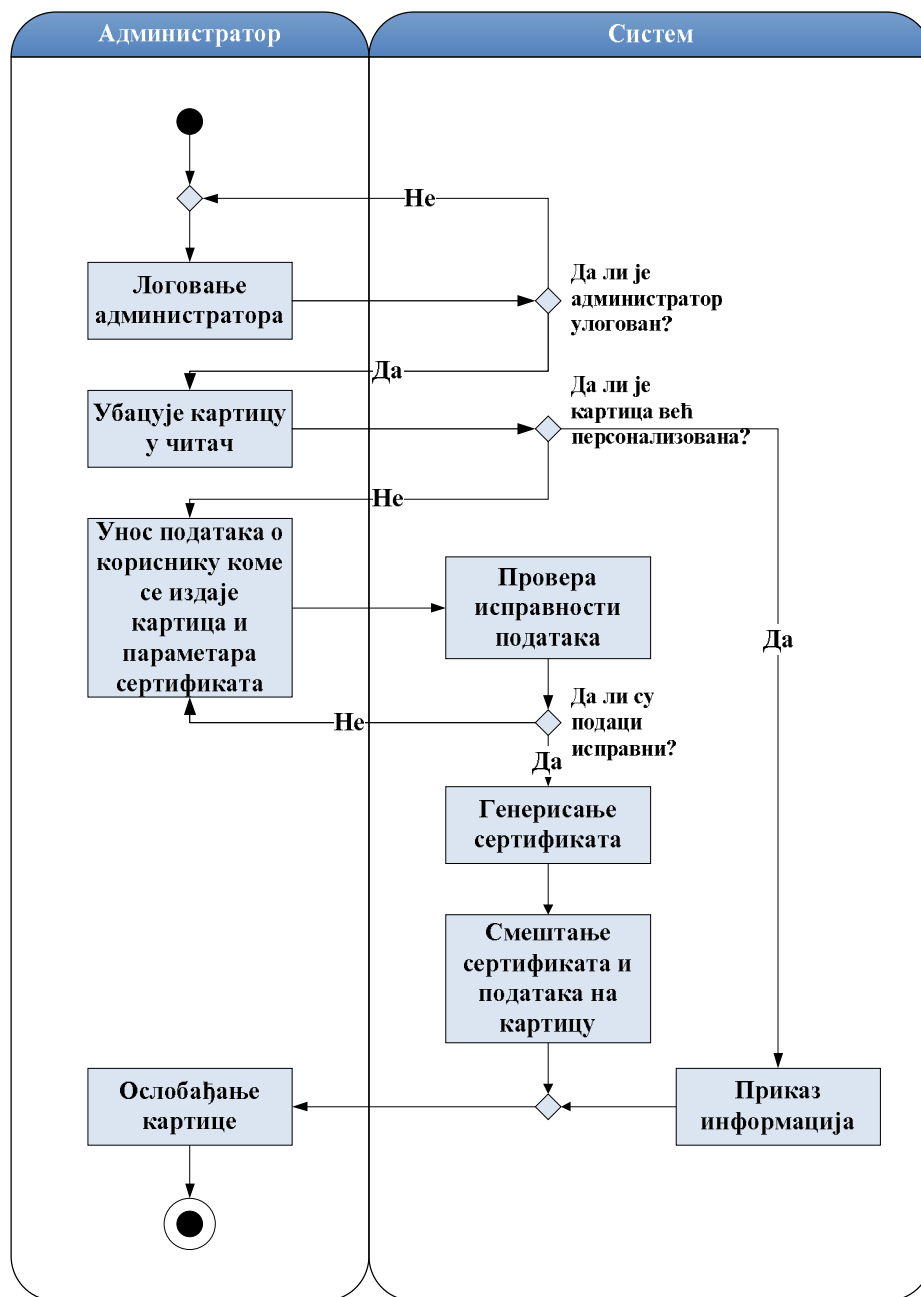
Класа **MyDigitalCertificate** садржи варијабле и методе потребне за преузимање, читање и креирање дигиталног сертификата.

MySmartCard је класа за интеракцију са смарт картицом и смештање дигиталних сертификата на исту. Њом су имплементирани методе којима се одређује стање читача картица / терминала, као и методе које се тичу интеракције са смарт картицом, смештање и читање дигиталних сертификата са смарт картице.

За израду логичког модела софтверског решења употребљен је *UML* језик [52]. Интеракција између класа моделирана је дијаграмима интеракције. Дијаграмом секвенце је приказан редослед активности, а дијаграмом колаборације структурна организација објеката који учествују у интеракцији [53].

8.5. Генерисање дигиталног сертификата и персонализација картице

Персонализацијом картице је предвиђено генерисање профила корисника у виду дигиталног сертификата и смештање на картицу.

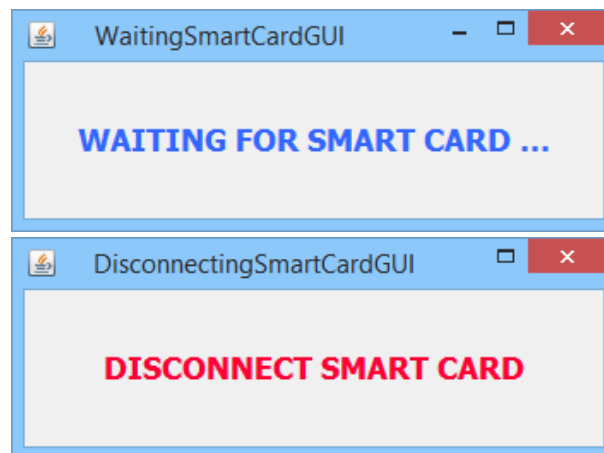


Дијаграм 8-5 UML дијаграм активности случаја употребе генерисања дигиталног сертификата и персонализације картице

Неопходни услови су да је администратор улогован за рад и да картица није већ персонализована. Уколико су испуњени наведени услови, администратор уноси податке о лицу коме се издаје сертификат и податке везане за сам сертификат. Доступна су различита подешавања сертификата који се креира као што су серијски број, одабир алгорита потписа и подешавање времена важења сертификата. Систем проверава да ли је администратор исправно унео податке и од њих креира дигитални сертификат.

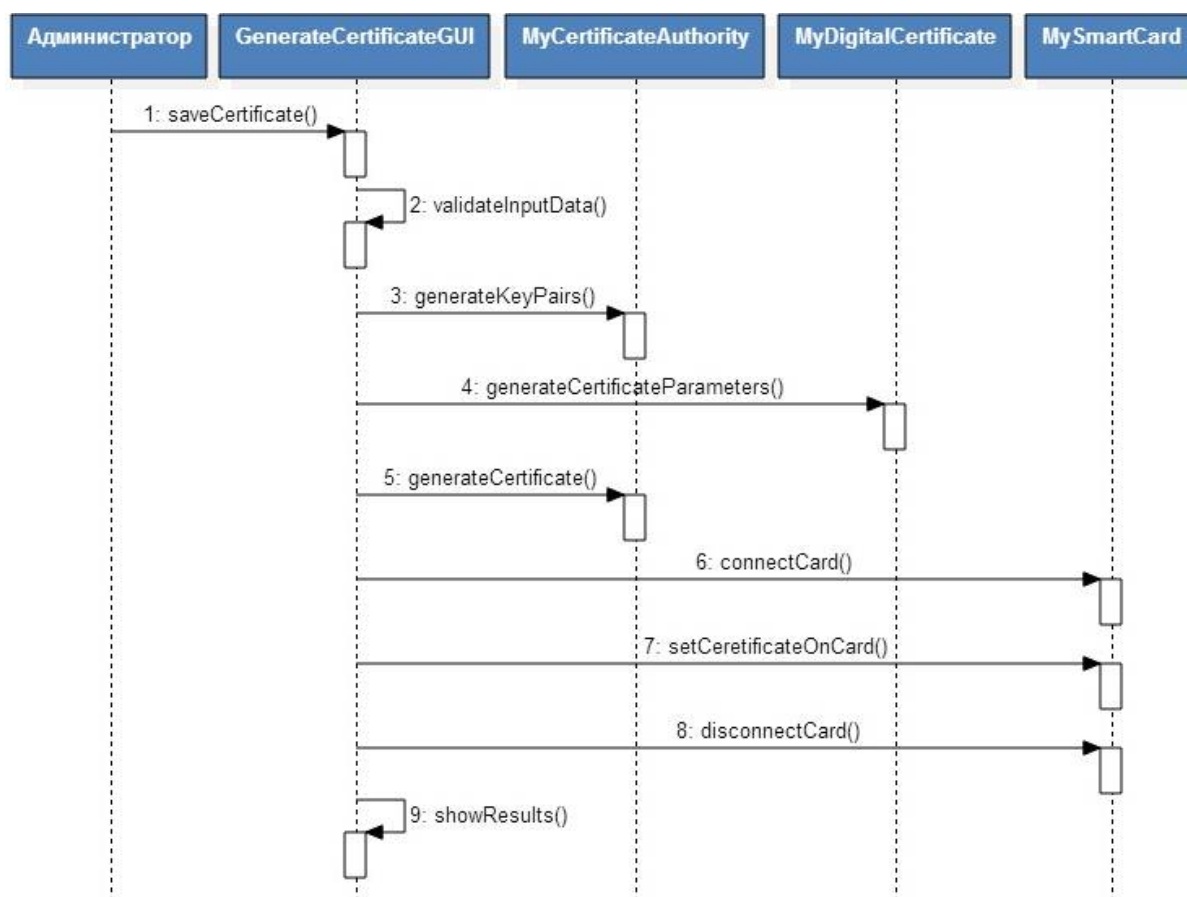
Слика 8-6 Графички кориснички интерфејс за генерисање сертификата и персонализацију картице

У било ком тренутку - приликом логовања, уколико дође до уноса података у супротности са предефинисаним правилима, уколико смарт картица није присутна или нестане конекција итд, приказују се информације у облику искачућих прозора. На тај начин се администратору даје повратна информација о захтеваним акцијама - да убади картицу, провери податке које је унео итд.



Слика 8-7 Прозори са обавештењима у корисничком интерфејсу

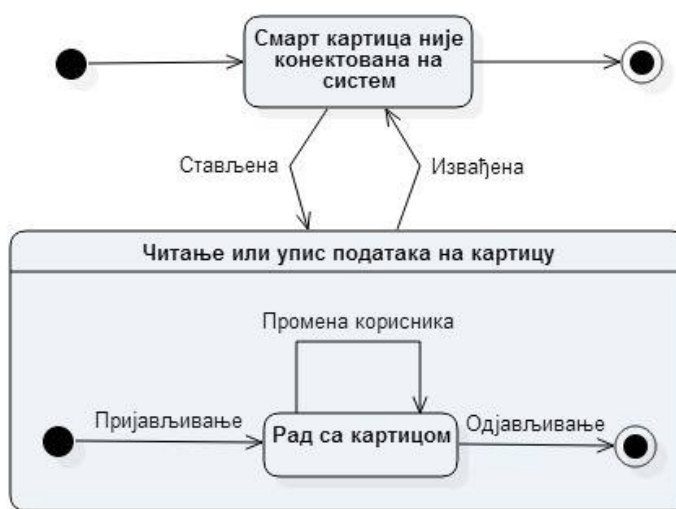
Приликом учитавања графичког корисничког интерфејса за генерисање сертификата и персонализацију картице - *GenerateCertificateGUI*, учитавају се приватни кључ и дигитални сертификат инстанце класе *CertificateAuthority*. Затим се врши генерисање пара асиметричних криптографских кључева за сертификат који се креира од стране класе *MyCertificateAuthority*. Потом се уносе параметри дигиталног сертификата. На крају се сертификат потписује приватним кључем инстанце класе *MyCertificateAuthority* и смешта се на картицу методама класе *MySmartCard*.



Дијаграм 8-8 UML секвенцијални дијаграм акције генерисања и смештања дигиталног сертификата на картицу

8.6. Аутентификација корисника

Дијаграм стања у којима систем може да се нађе у току рада са корисницима, приказан је на следећој слици. Систем или чека да се убади смарт картица или, ако је картица убачена, ради са подацима који су на њој [54].



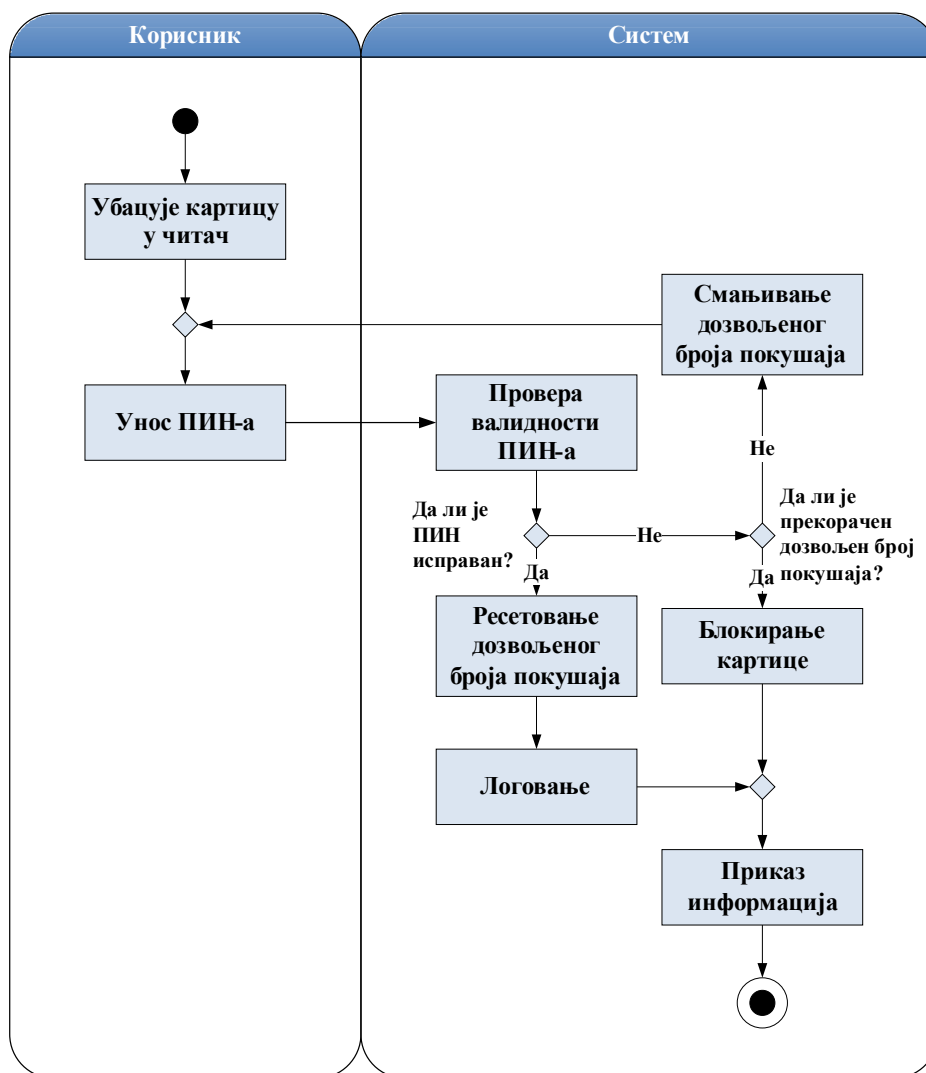
Дијаграм 8-9 UML дијаграм стања система

При аутентификацији (пријављивању) корисника помоћу смарт картице, неопходно је унети *PIN*. Затим се врши упоређивање унетог *PIN*-а са *PIN*-ом сачуваним на картици. Уколико је *PIN* идентичан, успешно је извршена аутентификација корисника и он може наставити даљи рад.



Слика 8-10 Прозор у корисничком интерфејсу са захтевом за уношење *PIN*-а

Сваки корисник приликом аутентификације има одређени максималан број покушаја да се аутентификује. Уколико он тај број покушаја премаше, картица се „закључава“ и корисник не може више да је користи. Картицу може да „откључа“ једино администратор. При успешној аутентификацији поново се уписује максималан број покушаја у систем.



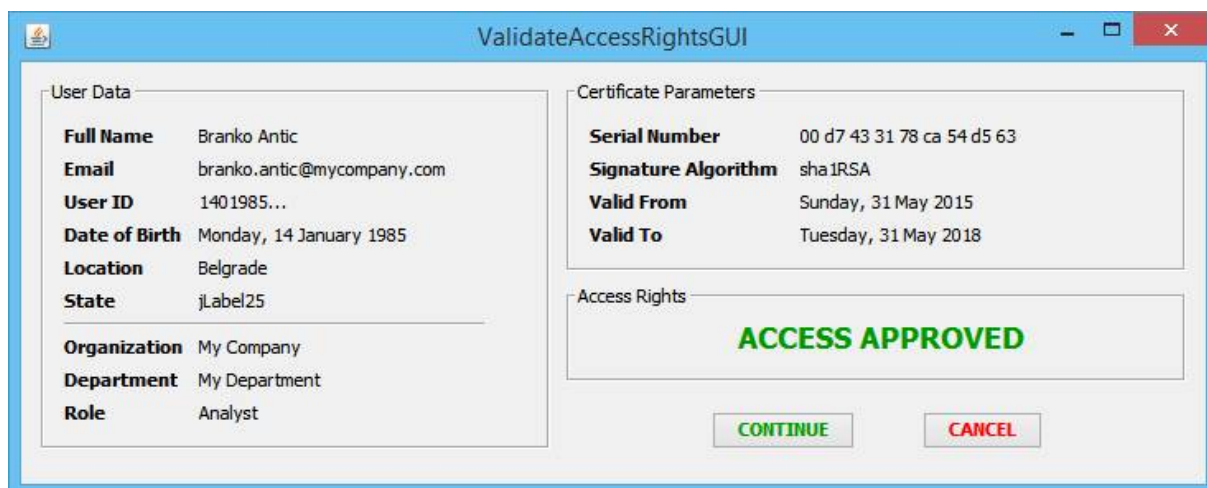
Дијаграм 8-11 UML дијаграм активности случаја употребе аутентификације корисника

8.7. Провера права приступа

Систем се налази у моду чекања картице. Детектовањем присутности картице, тражи потврду PIN-а. Затим се чита дигитални сертификат који садржи дефинисана права приступа корисника. Проверава се да ли је сертификат издат од стране одговарајућег сертификационог тела, да ли је валидан (није истекао) или је повучен. Подаци из сертификата и резултат провере права приступа се након тога исписују на графички кориснички интерфејс.

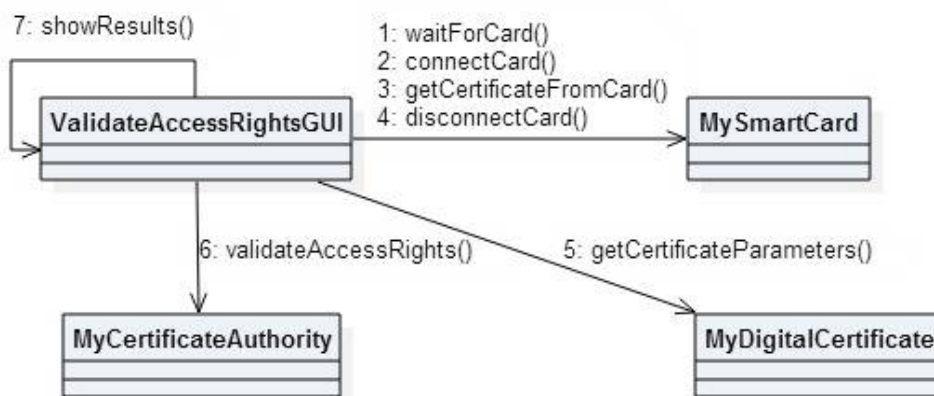


Дијаграм 8-12 UML дијаграм стања дела система намењеног за проверу права приступа

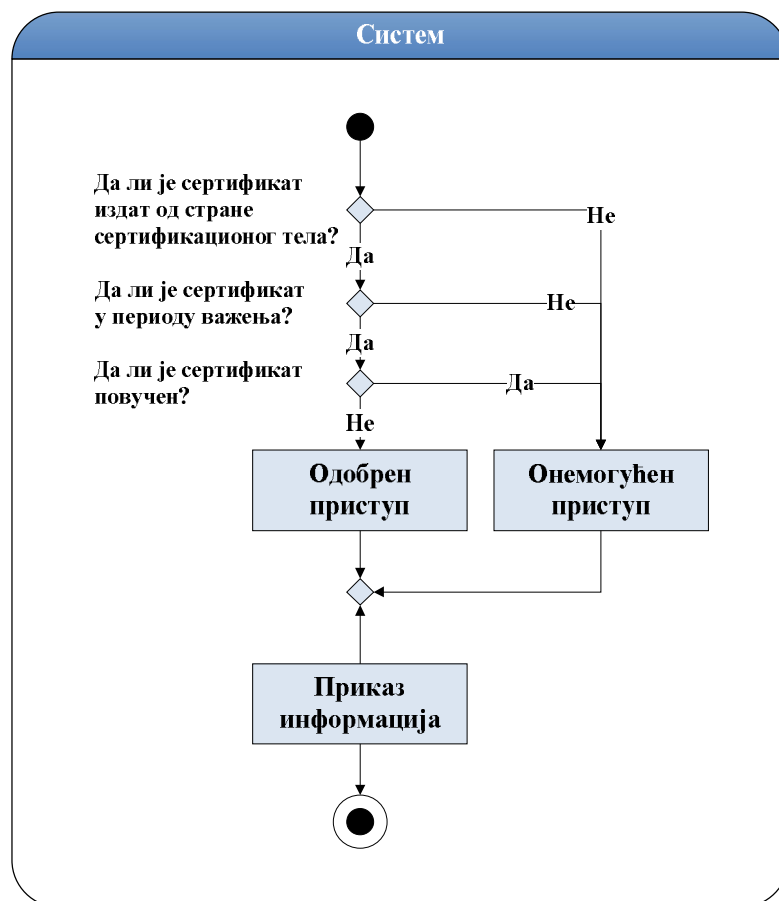


Слика 8-13 Изглед корисничког интерфејса за проверу права приступа када је картица присутна на терминалу

Детекцију присутности картице и читање параметара сертификата врши класа *MySmartCard*, док поверу права приступа врши *MyCertificateAuthority*. Прочитани дигитални сертификат и резултат провере се приказују на графичком интерфејсу имплементираним класом *ValidateAccessRightsGUI*.



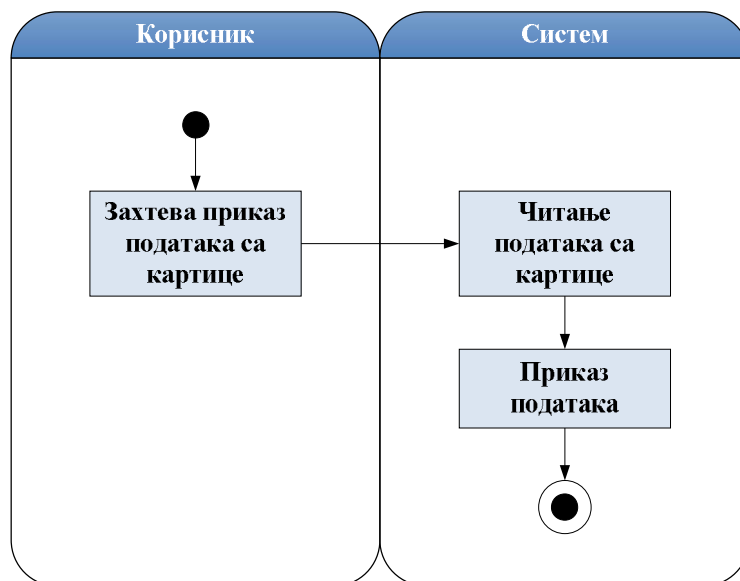
Дијаграм 8-14 UML дијаграм колаборације класа које учествују у провери права приступа



Дијаграм 8-15 UML дијаграм активности за проверу права приступа

8.8. Читање података са картице

Читање података са картице могу да врше сви корисници. Због тога није потребна аутентификација пре читања података. Тражени подаци се читају са смарт картице и шаљу као повратна вредност.

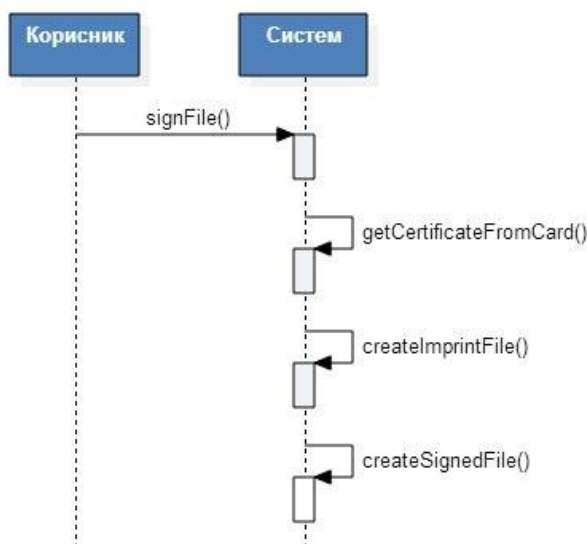


Дијаграм 8-16. UML дијаграм активности читања личних података са смарт картице

8.9. Дигитално потписивање података

Случај употребе поступка дигиталног потписивања обухвата активности којима се обезбеђује аутентичност лица које је дигитално потписало податке, интегритет тих података и немогућност да касније то лице порекне садржај потписаних података.

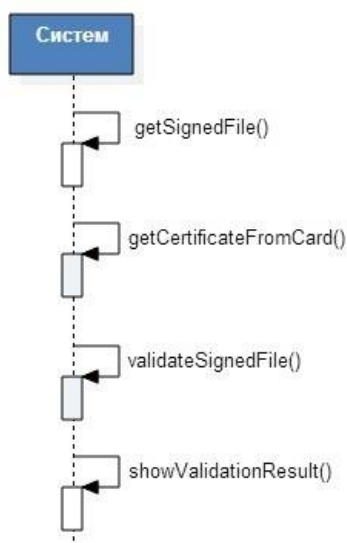
Случај употребе дигиталног потписивања података представља преузимање дигиталног сертификата са картице, чијим се тајним кључем потписује податак.



Дијаграм 8-17 Дијаграм секвенце дигиталног потписивања фајлова

8.10. Верификација дигитално потписаних података

Случај употребе верификације дигитално потписаних фајлова обухвата активности којима се утврђује веза између лица које је дигитално потписало податке и тих података, тј. утврђује се идентитет тог лица.



Дијаграм 8-18 Дијаграм секвенце верификације дигитално потписаних фајлова

9. Закључак

У раду је дат осврт на неке од облика угрожавања безбедности информационих систем.

Размотрени су начини употребе и валидације елемената *PKI* система (инфраструктура јавног кључа, дигитални потписи, дигитални сертификати, листе опозваних сертификата итд.) при реализацији криптографских технологија заштите тајности и аутентичности корисника и порука које се преносе у савременим информационим системима. Објашњен је начин функционисања целокупног система, као и део пратеће инфраструктуре овог система, са освртом на њену функционалност.

Приказана је и хијерархија сертификационих тела при успостављању дигиталног идентитета свих субјеката у оквиру дистрибуираног система, као и могућности примена смарт картица за потребе безбедности информационих система.

Обухваћени су основни концепти *Java* програмског језика и *Java Card* технологије, а који су коришћени у реализацији сопственог решења.

Циљ развоја апликације је да се обезбеди сервис за креирање и чување профила корисника на смарт картицама у облику дигиталног сертификата за потребе обезбеђења аутентичности, интегритета, непорецивости и заштите тајности у оквиру информационог система.

Коришћењем *UML* методологије дат је опис логичког модела апликације кроз примену пројектних образаца, са описом елемената који су били од суштинског значаја за имплементацију. Израђен је део графичког корисничког интерфејса који омогућава једноставно и сигурно коришћење аплета на смарт картици. Приликом реализације предложеног програмског решења, примењене су савремене методологије развоја објектно оријентисаног софтвера.

Правци даљег развоја апликације могу бити бројни, а условљени су првенствено могућношћу примене и конкретним потребама. Предлози смерница даљег рада су:

- реализација сопственог криптографског провајдера са сопственим имплементацијама симетричних и асиметричних криптографских алгоритама чиме би целокупан систем био независнији,
- смештање корисникових биометријских података на смарт картицу који би значајније одређивали његов идентитет,
- криптографска заштита базе података,
- повећање сигурности мрежних комуникација преко *SSL* протокола,
- инкорпорирање *POP3/SMTP* сервера за примање порука електронске поште,
- остале примене, у складу са конкретним потребама.

10. Списак слика, шема, дијаграма и табела

Шема 3-1 Симетрични криптографски алгоритми	- 8 -
Шема 3-2 Мод електронске кодне књиге	- 9 -
Шема 3-3 Мод уланчавања блокова	- 10 -
Шема 3-4 Мод повратног шифровања	- 10 -
Шема 3-5 Излазни повратни мод	- 11 -
Шема 3-6 IDEA алгоритам	- 12 -
Шема 3-7 Асиметрични криптографски алгоритамс	- 14 -
Шема 4-1 Хијерархијска структура сертификационих тела	- 19 -
Слика 4-2 Дигитални сертификат	- 20 -
Шема 4-3 Ланац дигиталних сертификата	- 20 -
Дијаграм 4-4 Поступак креирања и верификације дигиталног потписа	- 22 -
Шема 4-5 Структура дигиталног сертификата према X.509 стандарду	- 23 -
Табела 4-6 PKCS стандарди	- 25 -
Шема 4-7 TCP/IP мрежни модел и SSL протокол	- 26 -
Слика 5-1 Смарт картица	- 28 -
Слика 5-2 Контактна смарт картица	- 31 -
Слика 5-3 Бесконтактна смарт картица	- 32 -
Шема 5-4 Електронски контакти чипа смарт картице	- 32 -
Шема 5-5 Меморијски систем смарт картице	- 34 -
Слика 5-6 Уређаји за комуникацију са смарт картицама	- 36 -
Шема 5-7 Модел комуникације смарт картице и рачунара	- 37 -
Шема 5-8 Структура C-APDU	- 37 -
Шема 5-9 Структура R-APDU	- 37 -
Шема 5-10 Облици командних и одговарајућих APDU-а одговора	- 38 -
Шема 5-11 Структура фајл система смарт картице	- 39 -
Шема 5-12 Транспарентни фајл	- 39 -
Шема 5-13 Линеарни фајл са фиксном дужином слога	- 39 -
Шема 5-14 Линеарни фајл са променљивом дужином слога	- 39 -
Шема 5-15 Линеарни фајл са цикличном дужином слога	- 39 -
Табела 6-1 Наредбе keytool алата	- 44 -
Табела 6-2 Опције наредби keytool алата	- 45 -
Шема 7-1 Архитектура Java Card технологије	- 46 -
Шема 7-2 Структура Java Card Virtual Machine	- 47 -
Шема 7-3 Конвертовање пакета	- 49 -
Шема 7-4 Java Card Intaler и Off-Card инсталациони програм	- 50 -
Шема 7-5 Applet Identifier	- 51 -
Шема 7-6 Процес развоја аплета	- 52 -
Шема 8-1 Архитектура вишеслојне апликације	- 53 -
Дијаграм 8-2 UML дијаграм основних функционалности система у интеракцији са корисницима	- 54 -
Дијаграм 8-3 UML дијаграм основних пакета	- 55 -
Дијаграм 8-4 UML дијаграм класа у пакету mylogic	- 56 -
Дијаграм 8-5 UML дијаграм активности случаја употребе генерисања дигиталног сертификата и персонализације картице	- 57 -
Слика 8-6 Графички кориснички интерфејс за генерисање сертификата и персонализацију картице	- 58 -
Слика 8-7 Прозори са обавештењима у корисничком интерфејсу	- 58 -
Дијаграм 8-8 UML секвенцијални дијаграм акције генерисања и смештања дигиталног сертификата на картицу	- 59 -

Дијаграм 8-9 UML дијаграм стања система.....	- 59 -
Слика 8-10 Прозор у корисничком интерфејсу са захтевом за уношење PIN-а	- 60 -
Дијаграм 8-11 UML дијаграм активности случаја употребе аутентификације корисника	- 60 -
Дијаграм 8-12 UML дијаграм стања дела система намењеног за проверу права приступа	- 61 -
Слика 8-13 Изглед корисничког интерфејса за проверу права приступа када је картица присутна на терминалу	- 61 -
Дијаграм 8-14 UML дијаграм колаборације класа које учествују у провери права приступа	- 61 -
Дијаграм 8-15 UML дијаграм активности за проверу права приступа	- 62 -
Дијаграм 8-16. UML дијаграм активности читања личних података са смарт картице	- 62 -
Дијаграм 8-17 Дијаграм секвенце дигиталног потписивања фајлова	- 63 -
Дијаграм 8-18 Дијаграм секвенце верификације дигитално потписаних фајлова	- 63 -

11. Списак скраћеница

AES	- Advanced Encryption Standard
AID	- Applet Identifier
APDU	- Application Protocol Data Units
API	- Application Programming Interface
CA	- Certificate Authority
CAD	- Card Accepting Device
CAP	- Converted Applet
C-APDU	- Command Application Protocol Data Units
CBC	- Cipher Block Chaining Mode
CFB	- Cipher Feedback Mode
CISC	- Complex Instruction Set Computer
CLA	- Class of Instruction
CLK	- Clock
CN	- Common Name
CRL	- Certificate Revocation Lists
CSP	- Cryptographic Service Provider
CSR	- Certificate Signing Request
CSPRNG	- Cryptographically Secure Pseudo-Random Number Generator
DES	- Data Encryption Standard
DN	- Distinguished Name
DF	- Dedicated File
DSA	- Digital Signature Algorithm
DSS	- Digital Signature Standard
ECB	- Electronic Codebook Mode
ECDSA	- Elliptic Curve Digital Signature Algorithm
EEPROM	- Electrically Erasable Programmable Read-Only Memory
EF	- Elementary File
EMV	- Europay, MasterCard and Visa
ETSI	- European Telecommunications Standards Institute
EUR	- Euro
GND	- Ground
GSM	- Global System for Mobile Communications
GUI	- Graphical User Interface
HSM	- Hardware Security Module
HTTP	- HyperText Transfer Protocol
I/O	- Input / Output
IBM	- International Business Machines
IDEA	- International Data Encryption Algorithm
INS	- Instruction Code
ISO	- International Organization for Standardization
IP	- Internet Protocol
ISO-7816	- Identification Cards – Integrated Circuit Cards With Contacts
ISO-7816-3	- Electrical Interface and Transmission Protocols
ISO-7816-4	- Organization, Security and Commands for Interchange
JAR	- Java™ Archive
JCA	- Java Cryptography Architecture
JCE	- Java Cryptography Extension
JCRE	- Java Card Runtime Environment
JCVM	- Java Card Virtual Machine

JDK	- Java Development Kit
JIT	- Just-In-Time
JSSE	- JavaSecure Socet Extension
JVM	- Java Virtual Machine
MAC	- Message Authentication Code
MD	- Message Digest
MF	- Master File
MUA	- Mail User Agents
NIST	- National Institute of Standards and Technology
OFB	- Output Feedback Mode
OU	- Organisation Unit
PC	- Personal Computer
PCMCIA	- Personal Computer Memory Card International Association
PC/SC	- Personal Computer/Smart Card
PCBC	- Propagating Cipher Block Chaining
PIN	- Personal Identification Number
PIX	- Proprietary Identifier Extension
PKCS	- Public Key Cryptography Standard
PKI	- Public Key Infrastructure
POS	- Point of Sales
PVC	- Polyvinyl Chloride
RA	- Registration Authority
RAM	- Random Access Memory
R-APDU	- Response Application Protocol Data Units
RFC	- Request for Certificate
RID	- Resource Identifier
RIPEMD	- Race Interigy Primitives Evaluation
RISC	- Reduced Instruction Set Computer
ROM	- Read Only Memory
RSA	- Rivest, Shamir, Adleman algorithm
RST	- Reset
S/MIME	- Secure / Multipurpose Internet Mail Extension
SCQL	- Structured Card Query Language
SHA-1	- Secure Hash Algorithm
SIM	- Subscriber Identity Module
SRAM	- Static Random Access Memory
SSL	- Secure Sockets Layer
SW	- Status Word
TCP/IP	- Transmission Control Protocol / Internet Protocol
TLS	- Transport Layer Security
TPDU	- Transmission Protocol Data Units
TTP	- Trusted Third Party
UART	- Universal Asynchronous Receiver / Transmitter
UML	- Unified Modeling Language
USB	- Universal Serial Bus
VPN	- Virtual Private Network
XOR	- Exclusive Or

12. Литература

- [1] D.Avison & G.Fitzgerald, Information Systems Development: Methodologies, Techniques and Tools, Uxbridge: Brunel University, 2003.
- [2] R.C.Daigle, An Analysis of the Computer and Network Attack Taxonomy, Ohio: Wright-Patterson Air Force Base, 2001.
- [3] B.A.Forouzan, Data Communications and Networking, The McGraw-Hill, 2004.
- [4] M.E.Whitman & H.J.Mattord, Management of Information Security, Course Technology, 2008.
- [5] A.Menezes, P.van Oorschot & S.Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
- [6] М.Марковић, Технике заштите података и криптографски протоколи у савременим рачунарским мрежама, Београд, 2004.
- [7] B.Schneier, Applied Cryptography, New York: John Wiley & Sons, 1996.
- [8] L.R.Knudsen & M.J.B.Robshaw, The Block Cipher Companion, Springer Science & Business Media, 2011.
- [9] Y.Peng-Yeng, Trends in Developing Metaheuristics, Algorithms, and Optimization Approaches, IGI Global, 2012.
- [10] D.Stinson, Cryptography, Theory and Practice, Florida: CRC Press Inc, 1996.
- [11] Г.Ђорђевић, Реализација криптографских функција на копроцесорском модулу на бази дигиталног сигнала процесора, Београд, 2002.
- [12] М.Веиновић, М.Ћајић и Б.Бркић, Управљање криптолошким кључевима, Научни скуп - Синергија, 2011.
- [13] M.Mogollon, Cryptography and Security Services: Mechanisms and Applications, IGI Global, 2008.
- [14] H.C.A.Tilborg & S.Jajodia, Encyclopedia of Cryptography and Security, Springer Science & Business Media, 2011.
- [15] C.Adams & S.Lloyd, Understanding PKI: Concepts, Standards and Deployment Considerations, Addison-Wesley, 2003.
- [16] B.Ballad, T.Ballad & E.Banks, Access Control, Authentication and Public Key Infrastructure, Jones & Bartlett Publishers, 2010.
- [17] J.Feghhi, J.Feghhi & P.Williams, Digital Certificates: Applied Internet Security, Addison-Wesley, 1999.
- [18] М.Веиновић и С.Адамовић, Криптологија I: Основе за анализу и синтезу шифарских система, Београд: Унверзитет Сингидунум.
- [19] „X.509 Certificates and CRLs,“ 2015.
<http://docs.oracle.com/javase/8/docs/technotes/guides/security/cert3.html>.
- [20] R.Housley, W.Ford, W.Polk & D.Solo, RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999.
- [21] „Public Key Cryptography Standards,“ 2015. <https://en.wikipedia.org/wiki/PKCS>.
- [22] J.Steinberg, Understanding SSL VPN, 2005: Packt Publishing.
- [23] R.Oppliger, Secure Messaging with PGP and S/MIME, Artech House, 2001.
- [24] Smart Cards: A Case Study, IBM, 1998.
- [25] W.Effing & W.Rankl, Smart Card Handbook, Munich: John Wiley & Sons, 2003.
- [26] T.M.Jurgensen & S.B.Guthery, Smart Cards: The Developer's Toolkit, Prentice Hall

Professional, 2002.

- [27] A.Silberschatz, Operating Systems Concepts, Addison-Wesley, 1998.
- [28] „ISO-7816 Smart Card Standard,“ 2015.
http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx.
- [29] „GSM,“ 2015. <https://en.wikipedia.org/wiki/GSM>.
- [30] „EMVCo,“ 2015. <http://www.emvco.com/specifications.aspx>.
- [31] „Open Platform - Smart Card Standards and Specifications,“ 2015.
<http://www.smartcardalliance.org/smart-cards-intro-standards/>.
- [32] „OpenCard Framework,“ 2015. <http://www.openscdp.org/ocf/>.
- [33] „PC/SC Workgroup Specification,“ 2015.
<http://www.pcscworkgroup.com/specifications/overview.php>.
- [34] Б.Милосављевић, Практикум за курс - Java и интернет програмирање, Београд: Војна академија, 2001.
- [35] „Just-In-Time Compilation and Optimization,“ 2015.
http://docs.oracle.com/cd/E15289_01/doc.40/e15058/underst_jit.htm.
- [36] M.Pistoia, D.F.Reller, D.Gupta, M.Nagnur & A.K.Ramani, „Java 2 Network Security,“ International Technical Support Organization, 1999.
- [37] „Java Virtual Machine Specification,“ 2015.
<https://docs.oracle.com/javase/specs/jvms/se7/html/>.
- [38] J.Knudsen, Java Cryptography, O'Reilly, 1998.
- [39] „Java PKCS Reference Guide,“ 2015.
<http://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html>.
- [40] M.Pistoia, N.Nagaratnam, L.Koved & A.Nadalín, Enterprise Java Security: Building Secure J2EE Applications, Addison-Wesley, 2004.
- [41] „Java Secure Socket Extension,“ 2015.
<http://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html>.
- [42] „Java Cryptography Architecture,“ 2015.
<http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html>.
- [43] „Keytool - Key and Certificate Management Tool,“ 2015.
<http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html>.
- [44] „Java Card Technical Documentation,“ 2015.
<http://www.oracle.com/technetwork/java/embedded/javacard/documentation/javacard-docs-1970421.html>.
- [45] Z.Chen, Java Card Technology for Smart Cards: Architecture and Programmer's Guide, Addison-Wesley, 2000.
- [46] „Java Card Runtime Environment Specification,“ 2015.
<http://www.oracle.com/technetwork/java/javame/javacard/download/specs-138637.html>.
- [47] „Java Card Virtual Machine Specification,“ 2015.
<http://www.oracle.com/technetwork/java/embedded/javacard/documentation/javacard-docs-1970421.html>.
- [48] U.Hansmann, M.S.Nicklous, T.Schäck, A.Schneider & F.Seliger, Smart Card Application Development using Java, Berlin: Springer-Verlag, 2002.
- [49] D.Mazurek, uPortal and the Yale Central Authentication Service, Yale , 2004.
- [50] J.Centafont & B.Couillard, PKCS#11 Cryptographic Token Interface Standard, RSA laboratories, 2004.
- [51] „Java Tutorial,“ 2015. <https://docs.oracle.com/javase/tutorial/>.

- [52] G.Booch, J.Rumbaugh & I.Jacobson, The Unified Modeling Language - User Guide, Addison-Wesley, 1999.
- [53] Д.Милићев, М.Зарић и Н.Пироћанац, Објектно оријентисано моделовање на језику UML, Београд: Микро књига, 2001.
- [54] „Java Authentication and Authorization Service Reference Guide,“ 2015.
<http://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html>.