POLITECHNIKA POZNAŃSKA
WYDZIAŁ INFORMATYKI
INSTYTUT INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Legitymacja studencka z biometryczną identyfikacją osoby w ogólnopolskim systemie ochrony zdrowia OSOZ

Aplet jOSOZ

inż. Maciej Sobkowiak

Promotor:
prof. dr hab. inż. Joanna Józefowska

Opiekun:
mgr inż. Marek Gosławski

Poznań, 2014

Spis treści

1 Wstęp **3**

1.1 Wprowadzenie. 3
1.2 Technologie kart inteligentnych na przykładzie Elektronicznej Legitymacji Studenckiej 4
1.3 Technologie biometryczne 4
1.4 Ogólnopolski System Ochrony Zdrowia oraz funkcjonalność karty OSOZ . 5
1.5 Cel i zakres pracy 6

2 Techniczne aspekty kart inteligentnych oraz ich standaryzacja **9**

2.1 Komponenty fizyczne oraz protokół komunikacyjny 9
2.2 Systemy operacyjne i język JavaCard 11
2.3 Adaptacja maszyny wirtualnej Java dla kart inteligentnych 12
2.3.1 Zarządzanie pamięcią 12
2.3.2 Podstawowe biblioteki i ich charakterystyka 13
2.3.3 Biblioteki kryptograficzne 13
2.4 Standaryzacja użycia kart inteligentnych 14
2.4.1 Standardy ISO7816. 14
2.4.2 Standard GlobalPlatform 15

3 Biometria **17**

3.1 Model matematyczny cech biometrycznych 17
3.1.1 Matematyczna reprezentacja cech odcisku palca na postawie standardu ISO/IEC 19794-2:2005 17
3.1.2 Model matematyczny zastosowany w projekcie 18
3.2 Biometria w kontekście kart inteligentnych 19
3.2.1 Standaryzacja funkcji biometrycznych 19
3.2.2 Algorytm Match-on-card 20
3.2.3 Interfejs programistyczny realizujący funkcje związane z biometrią . 21

4 Projekt jOSOZ **23**

4.1 Etap projektowy. 23
4.1.1 Wymagania funkcjonalne 23
4.1.2 Wymagania pozafunkcjonalne 24

4.1.3 Przypadki użycia . . . 25
4.1.4 Architektura systemu . . . 25
4.1.5 Natywne funkcje apletu . . . 26
4.1.6 Funkcje zgodne z normą ISO7816-4 . . . 30
4.1.7 Obsługiwane polecenia standardu GlobalPlatform 2.1.1 . . . 32
4.1.8 Diagram klas apletu oraz ich opis . . . 32
4.1.9 Aplikacja na komputery PC prezentująca możliwości apletu . . . 34
4.1.10 Platforma deweloperska . . . 34
4.2 Implementacja . . . 35
4.2.1 Realizacja systemu plików . . . 35
4.2.2 Mechanizm dostępu do plików . . . 36
4.2.3 Obsługa PIN-u . . . 36
4.2.4 Wykorzystane funkcje i obiekty kryptograficzne . . . 37
4.2.5 Implementacja klasy obiektu biometrycznego . . . 37
4.2.6 Algorytm Match-on-card . . . 40
4.2.7 Kod OSOZ i podpisywanie wiadomości . . . 41
4.3 Testy . . . 42
4.3.1 Testy poprawności implementacji funkcji apletu . . . 42
4.3.2 Skuteczność algorytmu weryfikującego odcisk palca . . . 42
4.4 Bezpieczeństwo proponowanego rozwiązania . . . 45

5 Wnioski i zalecenia wdrożeniowe 47

A Przypadki użycia 49

B Płyta CD 55

Bibliografia 57

Rozdział 1

Wstęp

1.1 Wprowadzenie

Karty inteligentne, których historia sięga lat 50. ubiegłego wieku na dobre zagościły w wielu dziedzinach ludzkiego życia [1]. Ich niewielki rozmiar pozwala każdemu na noszenie ich przy sobie. Prostota użycia pozwala na użycie ich przez osoby niemające pojęcia o nowoczesnych technologiach.

Pomimo bardzo ograniczonych możliwości sprzętowych znajdują one bardzo szerokie zastosowanie. Przykładami mogą być:

- karty SIM wykorzystywane w telefonii GSM,
- karty płatnicze wydawane przez praktycznie każdy bank,
- elektroniczna portmonetka,
- programy lojalnościowe,
- nośnik certyfikatu,
- podpisy elektroniczne,
- bilety komunikacji miejskiej,
- uwierzytelnianie i autoryzacja do zasobów w systemach informatycznych,
- systemy fizycznej kontroli dostępu do pomieszczeń.

Wraz z wygodą użycia karty elektroniczne oferują także wysoki poziom bezpieczeństwa. W głównej mierze zależy on od fizycznego dostępu do karty. Wiele definicji i ogólnych opisów możliwych ingerencji w kartę można znaleźć w pozycji [2] oraz [1]. Typowe ataki, które są stosowane w celach przestępczych bez próby przejęcia blankietu karty polegają głównie na skopiowaniu paska magnetycznego karty oraz ewentualnym odczycie kodu autoryzującego. Obecne paski magnetyczne są coraz rzadziej obecne we współczesnych kartach i zostały w pełni zastąpione o wiele lepiej zabezpieczonym interfejsem procesorowym. Dostęp do pamięci chroniony jest przez system operacyjny karty i mechanizmy sprzętowe zastosowanego mikroprocesora.

Rozwiązania kartowe stosowane w różnych systemach ciągle ewoluują, wprowadzane są coraz to nowsze funkcjonalności. Jedna karta może pełnić wiele funkcji w kilku systemach. Problematyka pracy będzie dotyczyć rozszerzenia funkcji konkretnego rozwiązania

kartowego o obsługę funkcji w zewnętrznym systemie, opracowanym przez dużą firmę informatyczną.

1.2 Technologie kart inteligentnych na przykładzie Elektronicznej Legitymacji Studenckiej

Praktycznym przykładem wykorzystania technologii kart inteligentnych w życiu codziennym może być system Elektronicznej Legitymacji Studenckiej. Wprowadzony został on dzięki wydaniu rozporządzenia Ministra Nauki Szkolnictwa Wyższego z dnia 2 listopada 2006 r. *w sprawie dokumentacji przebiegu studiów* [26]. Rozporządzenie to wprowadzało możliwość zmiany tradycyjnej papierowej legitymacji studenckiej na elektroniczne blankiety z układem elektronicznym, które posiadałyby odpowiedni, zgodny z ustawowymi wymogami, nadruk. Stemple przedłużające ważność takiej legitymacji zamienione zostały na trudniejsze do podrobienia hologramy. Dodatkowo ważność takiej legitymacji przedłuża się także elektroniczne poprzez modyfikację zawartości pamięci karty. W przypadku podejrzenia fałszerstwa dowolnego elementu legitymacji (hologram, nadruk na blankiecie), który weryfikowany jest bez pomocy specjalistycznych czytników istnieje możliwość weryfikacji elektronicznej legalności dokumentu.

Legitymacja, oprócz tradycyjnej funkcji jaką jest poświadczenie statusu studenta, dzięki przejściu na nośnik elektroniczny może pełnić typowe funkcje kartowe. Dla miasta Poznań ELS integruje w sobie między innymi [3]:

- elektroniczny bilet komunikacji miejskiej,
- aplikację biblioteczną,
- kod kreskowy do uniwersalnego zastosowania,
- identyfikator w systemach kontroli dostępu do pomieszczeń (niektóre uczelnie),
- nośnik certyfikatu, umożliwiającego korzystanie z różnych funkcjonalności uczelnianych systemów informatycznych,
- rejestracja wejść lub obecności.

Funkcjonalność ELS może być rozbudowywana poprzez implementację własnego oprogramowania przeznaczonego na karty inteligentne. Każda z tych aplikacji może pełnić inną funkcję i współpracować z różnymi systemami. Dzięki temu każdy student może dokonywać różnych operacji posiadając tylko i wyłącznie jedną, wielofunkcyjną kartę procesorową.

1.3 Technologie biometryczne

Historia zastosowań biometrii, które znamy z obecnych czasów, sięga końca XIX wieku [27]. W ogólności dziedzina ta zajmuje się pobieraniem indywidualnych cech biologicznych organizmu, matematycznym modelowaniem oraz ich porównywaniem. Każdy żywy organizm posiada unikalny zestaw takich cech, który pozwala na jego jednoznaczną identyfikację. Rozwój informatyki pozwolił na zastosowanie biometrii w wielu dziedzinach

ludzkiej działalności. Obecnie istnieje wiele modułów oferujących biometryczną weryfikację poprzez:

- odciski palców,
- obraz siatkówki oka,
- układ naczyń włosowatych znajdujący się pod skórą,
- obraz lub geometrię twarzy,
- cechy głosu,
- charakter pisma.

Gwałtowny rozwój zastosowania biometrii nastąpił w końcówce lat sześćdziesiątych ubiegłego wieku, kiedy amerykańskie Federalne Biuro Śledcze zaczęło finansować opracowanie technologii systemów automatycznej identyfikacji odciskami palców (AFIS). Pozwalały one na automatyczne dopasowanie zabezpieczonych linii papilarnych do osób wcześniej notowanych za przestępstwa kryminalne, których odciski palców na stałe wprowadzone zostały do bazy danych dostępnej dla wszystkich jednostek policji. Pozwalały również na udowodnienie zatrzymywanym osobom przestępstw, które do tej pory nie zostały wykryte.

Aby dokonać procesu porównania cech biometrycznych, próbka biometryczna musi przejść szereg operacji:

- odpowiednie zabezpieczenie lub jej pobranie - celem tego etapu jest stworzenie danych wejściowych (najczęściej pliku graficznego) dla modułów tworzących model matematyczny cechy,
- wyodrębnienie cech charakterystycznych próbki z danych wejściowych - przekształcenie do postaci dostosowanej dla urządzeń elektronicznych
- utworzenie wzorca biometrycznego - jest to zbiór danych wejściowych dla algorytmu porównującego cechy,
- porównanie i zwrócenie podobieństwa próbek - wynik działania algorytmu operującego na utworzonych wzorca biometrycznych.

Weryfikacja biometryczna znajduje szereg zastosowań w życiu codziennym, między innymi w kontroli dostępu do pomieszczeń, bankomatach, dostępie do zasobów informatycznych. Stosowanie biometrii z innymi metodami autoryzacji pozwala na tworzenie systemów o wysokich gwarancjach bezpieczeństwa danych.

W niniejszej pracy opracowano i zaimplementowano autorski wzorzec biometryczny oraz algorytm porównujący cechy z wykorzystaniem linii papilarnych człowieka.

1.4 Ogólnopolski System Ochrony Zdrowia oraz funkcjonalność karty OSOZ

Ogólnopolski System Ochrony Zdrowia (OSOZ) to specjalnie przygotowana platforma informatyczna wspierająca procesy medyczne dostępna dla każdego obywatela Polski [28]. System ten pozwala na założenie własnego konta zdrowotnego, które oferuje rejestrowanie szeregu informacji o zdrowiu pacjenta:

- przebyte choroby i dolegliwości,
- zażywane leki lub przepisane recepty,
- historia zabiegów,
- wyniki badań.

Dane z punktu widzenia systemu są w pełni anonimowe. Pacjent dobrowolnie przekazuje je lekarzowi, który może śledzić historię zdrowia pacjenta. Dostęp do takich danych odbywa się z reguły przez kartę pacjenta, zawierającą unikalny identyfikator jej posiadacza w systemie. Po uwierzytelnieniu się kodem PIN lub poprzez weryfikację biometryczną przez pacjenta w gabinecie lekarskim współpracującym z OSOZ lekarz może odczytać zapisane na koncie informacje i wykorzystywać je w celach medycznych.

Procesorowa karta OSOZ udostępnia następujące funkcje [5]:

- weryfikacja kodem PIN,
- weryfikacja biometryczna,
- generowanie kodu autoryzacyjnego,
- podpisywanie świadczeń medycznych,
- identyfikacja pacjenta poprzez numer zapisany na karcie,
- przechowywanie danych wraz z kontrolą dostępu ich odczytu.

Za pomocą tej karty możliwe jest pełne wykorzystanie funkcji dostępnych dla pacjenta oferowanych przez platformę OSOZ.

1.5 Cel i zakres pracy

Celem projektu była implementacja funkcjonalności identyfikujących oraz autoryzujących użytkownika udostępnianych przez kartę elektroniczną, która została opracowana przez firmę KAMSOFT. Projekt ten dedykowany jest dla powszechnie wydawanych elektronicznych legitymacji studenckich. Pozwoli to każdemu studentowi, posiadającemu zaprojektowany aplet, na dostęp do tegoż systemu z wykorzystaniem posiadanej karty. Podobne rozwiązanie stosowane jest na przykład w przypadku integracji biletów komunikacji miejskiej w Poznaniu z ELS [3] lub rozszerzenie funkcji ELS o uniwersalny identyfikator biblioteczny [4]. Wynikami końcowymi pracy są:

- aplet napisany w języku JavaCard, implementujący funkcje karty użytkownika systemu OSOZ,
- aplikacja przeznaczona na komputery typu PC, demonstrująca możliwości apletu,
- dokumentacja, pozwalająca na późniejszą integrację rozwiązania z innymi systemami i jego modernizację,

Niniejsza praca magisterska złożona z 5 rozdziałów. W rozdziale drugim zawarto krótki opis technologii kart inteligentnych i normy regulujące implementację interfejsów programistycznych karty wraz z referencjami do innych prac. Rozdział trzeci opisuje teoretyczne podstawy wykorzystania biometrii w informatyce, jej zastosowanie w niniejszym projekcie

oraz szczegółowo omawia założenia opracowanego rozwiązania. Rozdział czwarty przedstawia całość etapu projektowego oraz wszystkie etapy jego wykonania. Końcowy rozdział zawiera wnioski autora oraz zalecenia wdrożeniowe dla odbiorcy projektu.

Rozdział 2

Techniczne aspekty kart inteligentnych oraz ich standaryzacja

2.1 Komponenty fizyczne oraz protokół komunikacyjny

Karty inteligentne cechują się prostą budową fizyczną. Najważniejszym ich elementem są układy elektroniczne realizujące podstawowe operacje. W przypadku kart procesorowych są one widoczne w postaci pozłacanych styków. Ich fizyczny kontakt z czytnikiem kart dostarcza zasilania i pozwala wykonywać podstawowe operacje. Dla kart bezstykowych najważniejszą częścią jest antena połączona z procesorem, która po zbliżeniu do czytnika wytwarza prąd na zasadzie indukcji elektromagnetycznej. Procesor realizujący funkcjonalność kart MIFARE jest do anteny bezpośrednio podłączony, co pozwala na odbiór transmisji oraz zasilenie układu.

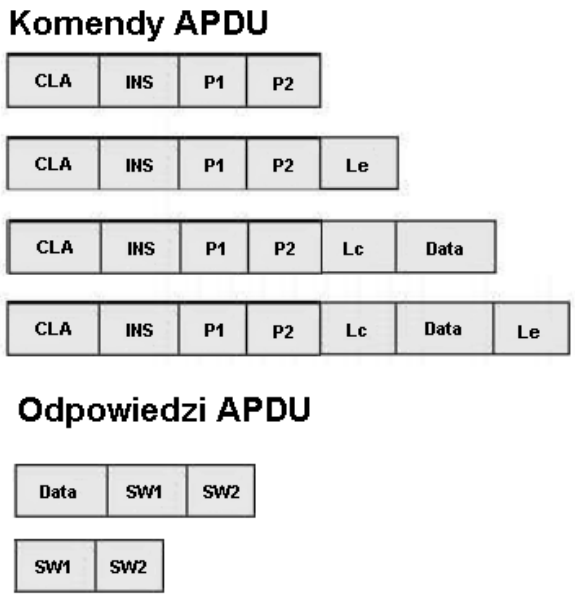
Mikroprocesory użyte w kartach inteligentnych integrują w sobie pamięć (zarówno trwałą - EEPROM jak i nietrwałą - RAM). Często integrują w sobie dodatkowy układ, dedykowany do operacji kryptograficznych (z racji częstego stosowania kart jako nośnik kluczy, wymagane także do bezpiecznej komunikacji z kartą).

Wszystkie układy elektroniczne zatopione są w plastiku. Wymiary tak przygotowanych kart określa norma ISO7810. Dokładniejsze informacje o wymiarach określanych przez tą normę i rodzajach wytwarzanych kart można znaleźć w [1], [2], [29].

Rysunek 2.1: Typowa karta inteligentna

Komunikacja karty z czytnikiem odbywa się poprzez protokół APDU. W protokole tym wyróżnia się dwa rodzaje komunikatów:

- komenda APDU wysyłana do karty, zawiera obowiązkowo minimum 4 bajty określające rodzaj komendy i parametry jej wykonania. Dodatkowo może zawierać długość pola danych, dane przesyłane do karty oraz liczbę bajtów oczekiwanych w odpowiedzi,
- odpowiedź APDU zwracana przez kartę. Dwa ostatnie bajty zawierają status odpowiedzi. Dodatkowo przed statusem odpowiedzi zwrócone mogą zostać bajty zawierające dane zwrotne od karty.



Rysunek 2.2: Możliwe warianty poleceń i odpowiedzi APDU

Poszczególne pola oznaczają odpowiednio:

- **CLA** - klasa instrukcji dla wysyłanej komendy, określone klasy instrukcji są zdefiniowane w standardach ISO,
- **INS** - bajt oznacza konkretną instrukcję z klasy do wykonania przez kartę,
- **P1** i **P2** - definiują parametry wywołania polecenia,
- **Lc** - określa liczbę bajtów danych, które polecenie zawiera,
- **Data** - dane zwracane lub wysyłane do karty,
- **Le** - liczba bajtów oczekiwanych w odpowiedzi APDU,
- **SW1** i **SW2** - definiują status wykonania komendy APDU.

Maksymalna dozwolona długość pola danych dla większości kart wynosi 256 bajtów. Istnieją karty korzystające z ulepszonej wersji tego protokołu (*extended APDU*). W tym wypadku do karty można za pomocą jednej komendy przesłać aż 32768 bajtów (maksymalny zakres zmiennej typu short). Wykorzystane w tym projekcie karty nie wspierają jednak tego standardu. Opis fizycznej realizacji kanału komunikacyjnego pomiędzy kartą a czytnikiem wraz kontrolą integralności datagramów opisano dokładniej w [2], [29].

2.2 Systemy operacyjne i język JavaCard

Każde złożone urządzenie elektroniczne mogące pełnić zaawansowane operacje z reguły posiada zintegrowane oprogramowanie pozwalające na zarządzanie zasobami w nim zintegrowanych. Karty inteligentne są dobrym przykładem takich urządzeń. Dzięki systemowi operacyjnemu karty inteligentne pozwalają zapewnić między innymi:

- wieloaplikacyjność,
- implementację systemu plików,
- zarządzanie pamięcią trwałą i nietrwałą,
- przełączanie kontekstu między aplikacjami,
- mechanizmy transakcji dostępne na żądanie programisty,
- implementację i dostęp do modułów znajdujących się na karcie (np. natywne wsparcie dla biometrii, implementacja bezpiecznego kanału, obsługa globalnego kodu PIN, zarządzanie aplikacjami na karcie)
- współpracę między aplikacjami,
- utworzenie bezpiecznego środowiska aplikacyjnego,
- ochronę pamięci,
- API dostępne dla programisty

Do najbardziej znanych systemów kartowych systemów operacyjnych należą [2]:

- JavaCard,
- SmallOS,
- MULTOS,
- MPCOS.

W niniejszej pracy uwaga skupiona zostanie na systemie JavaCard, który jest adaptacją maszyny wirtualnej Java znanej z komputerów PC na karty elektroniczne. Pozwoliło to na użycie wysokopoziomowego języka programowania obiektowego do tworzenia przez inżynierów własnych aplikacji rozszerzających możliwości kart.

Język Java na platformie JavaCard posiada bardzo istotne ograniczenia dostosowane do możliwości sprzętowych oferowanych przez karty inteligetne:

- obsługa zmiennych całkowitoliczbowych *byte* oraz *short* wraz z obsługą liczb ujemnych, zmienne typu boolean,
- brak wielowątkowości,
- jednowymiarowe tablice,
- tworzenie obiektów zawierających metody i pola o typach wspieranych przez JavaCard,
- możliwość tworzenia obiektów w pamięci trwałej[1] oraz nietrwałej[2],

[1] Alokacja najczęściej następuje automatycznie.
[2] JavaCard oferuje dedykowane metody do tworzenia tablic w pamięci nietrwałej.

- prosty mechanizm transakcji oferujący niewielki bufor transakcji (*commit buffer*) i brak możliwości ich zagnieżdżania (*transaction nesting*),
- wsparcie dla kryptografii poprzez dostęp do zaimplementowanych na karcie mechanizmów.

W zależności od wersji JavaCard dostarczone mogą być dodatkowe biblioteki ułatwiające np. implementację systemu plików zgodnego z normą ISO7816-4 w tworzonej aplikacji lub oferujące wsparcie dla biometrii.

2.3 Adaptacja maszyny wirtualnej Java dla kart inteligentnych

2.3.1 Zarządzanie pamięcią

Do tworzenia oprogramowania karty inteligentne wyposażone w system operacyjny JavaCard oferują dwa rodzaje pamięci, które służą do składowania danych:

- pamięć nietrwała (RAM) czyszczona po każdym resecie karty lub deselekcji aplikacji
- pamięć trwała (EEPROM) która przechowuje dane w obrębie aplikacji po wyłączeniu zasilania

Programując w języku Java na kartach inteligentnych należy stosować się do zasad organizacji pamięci i przydzielania jej w zależności od stosowanych instrukcji. Nieprawidłowe korzystanie z API może doprowadzić do sytuacji, w której zabraknie pamięci trwałej karty do wykonywania operacji. W tej sytuacji jedyną możliwością zwolnienia pamięci jest ponowna instalacja apletu. Pamięć trwała zajmowana jest w następujących sytuacjach:

- tworzenie obiektów operatorem *new*
- deklarowanie globalnych zmiennych
- używanie zmiennej globalnej do adresowania tablic
- deklarowanie stałych poprzez słowa kluczowe *static final*

Pamięć nietrwała (RAM) jest w kartach bardzo ograniczona. Przekroczenie limitu jej użycia powoduje najczęściej konieczność poprawy implementacji metody z niej korzystającej. Używana jest między innymi w przypadku:

- wywoływania metod i przekazywania parametrów
- tworzenia zmiennych lokalnych metody
- jawnego tworzenia obiektów przy pomocy metod klasy JCSystem (np. *makeTransientByteArray*)

Pamięć nietrwała dzieli się na dwa rodzaje, które są alokowane w osobnych obszarach:

- czyszczona przy resecie karty (oznaczana stałą CLEAR_ON_RESET)
- czyszczona przy deselekcji apletu (oznaczana stałą CLEAR_ON_DESELECT)

Istnieje możliwość sprawdzenia stanu dostępnej ilości pamięci każdego z wymienionych typów z wykorzystaniem klasy JCSystem, jednak w przypadku pamięci trwałej może zdarzyć się sytuacja, w której zwrócona wartość będzie mniejsza niż faktyczna liczba dostępnych bajtów z racji przekroczenia zakresu zmiennej *short*. Dodatkowo od wersji 2.1.2 dostępny jest *garbage collector* - mechanizm umożliwiający programowe usunięcie z pamięci RAM nieużywanych obiektów.

2.3.2 Podstawowe biblioteki i ich charakterystyka

Poniższe zestawienie przedstawia biblioteki dostarczane wraz z pakietem JavaCard 2.2.1:

- obsługa znanych wyjątków oraz obiektowości w pakiecie java.lang,
- interfejs do implementacji usług RMI z wykorzystaniem kart inteligentnych w pakietach java.rmi oraz javacard.framework.service,
- bogaty zestaw obiektów kryptograficznych zawarty w pakietach javacard.security i javacardx.crypto. Oferowane są popularne funkcje skrótów, mechanizmy szyfrowania symetrycznego i asymetrycznego. Dostarczane są także obiekty wspomagające konstrukcję i bezpieczne przechowywanie kluczy. Pakiet ten dostarcza także mechanizm generowania liczb pseudolosowych,
- podstawowe biblioteki pozwalające na wygodną obsługę poleceń APDU oraz dostarczających narzędzia do kopiowania tablic, wykorzystywania pamięci nietrwałej, obsługi transakcji, kodu PIN, interfejsu Shareable, interfejsu MultiSelectable i stałych charakterystycznych dla standardów kart inteligentnych zawarte w pakiecie javacard.framework.

2.3.3 Biblioteki kryptograficzne

Popularność kart inteligentnych jako urządzeń obsługujących procesy uwierzytelniania i nośnik certyfikatów sprawiła, że praktycznie każda karta inteligentna posiada wsparcie dla kryptografii. Dostawcy systemów operacyjnych dostarczają więc gotowe do wykorzystania metody i obiekty w pełni wykorzystujące potencjał sprzętowy wbudowanego koprocesora wspierającego funkcje kryptograficzne. Dla systemu JavaCard 2.2.1 dostępne są następujące mechanizmy:

- funkcje skrótów - obsługiwane przez klasę MessageDigest,
- szyfrowanie symetryczne i asymetryczne oferowane przez klasę Cipher,
- mechanizmy generowania i weryfikacji podpisów cyfrowych oferowany przez klasę Signature,
- mechanizmy budowania i wykorzystywania kluczy dostarczane między innymi przez klasę KeyPair oraz inne,
- wsparcie dla algorytmu uzgadniania kluczy Diffiego-Hellmana (klasa KeyAgreement) i mechanizm krzywych eliptycznych.

Do dostępnych do wykorzystania funkcji skrótu należą:

- SHA1,

- SHA256/384/512,
- MD5,
- RIPEMD160.

Wspierane szyfry symetryczne i asymetryczne w różnych trybach:

- AES CBC/EBC,
- DES/TripleDES CBC/ECB z różnymi wariantami *paddingu*,
- Korean Seed,
- RSA z różnymi wariantami *paddingu*.

Wspierane podpisy cyfrowe:

- AES MAC,
- DES/TripleDES (różne długości podpisu),
- DSA (różne tryby),
- HMAC (różne tryby),
- Korean Seed,
- RSA (różne tryby, długość zależna od użytego rozmiaru klucza).

Wspierane mechanizmy generowania kluczy:

- RSA - różne długości (od 512 bitów do 2048 bitów), generowanie pary (publiczny i prywatny),
- DSA - różne długości, generowanie pary (publiczny i prywatny),
- EC - różne tryby i długości, generowanie pary (publiczny i prywatny),
- AES,
- DES,
- HMAC,
- Korean Seed.

Do poprawnego działania wszystkich powyższych mechanizmów niezbędną jest ich odpowiednia implementacja na blankiecie karty dokonywana przez dostawców sprzętu.

2.4 Standaryzacja użycia kart inteligentnych

2.4.1 Standardy ISO7816

Szerokie zastosowania kart elektronicznych w życiu codziennym wymusiły powstanie dokumentów standaryzujących ich budowę oraz funkcje. Przestrzeganie standardów pozwala na wygodne używanie kart z każdym urządzeniem zgodnym z normami. Dokumenty opisujące standardy kart zawarte są w szeregu norm ISO7816 posiadającego 14 części:

- ISO7816-1 - określa własności fizyczne oraz wymiary kart,

- ISO7816-2 - określa wymiary oraz położenie styków dla interfejsu stykowego,
- ISO7816-3 - określa wymagane własności sygnałów elektrycznych przesyłanych do karty oraz własności protokołu transmisyjnego,
- ISO7816-4 - określa organizację danych wewnątrz karty, mechanizmy bezpieczeństwa oraz normuje podstawowe polecenia realizujące funkcje aplikacji,
- ISO7816-5 - określa kroki potrzebne do rejestracji dostawców aplikacji kartowych na ogólnoświatowym szczeblu
- ISO7816-6 - określa etykiety (*tags*), którymi oznaczane są często stosowane w kartach inteligentnych dane,
- ISO7816-7 - określa sposób implementacji poleceń bazodanowych (*SCQL - smart card query language*) w środowisku kart inteligentnych,
- ISO7816-8 - określa implementację poleceń związanych z kryptografią,
- ISO7816-9 - określa implementację poleceń zarządzających systemem plików karty (zarządzanie aplikacjami),
- ISO7816-10 - określa własności sygnałów elektrycznych dla kart implementujących synchroniczną transmisję danych z czytnikiem,
- ISO7816-11 - określa sposób implementacji funkcji biometrycznych na kartach inteligentnych,
- ISO7816-12 - określa sposób wykorzystania interfejsu USB na kartach inteligentnych,
- ISO7816-13 - określa implementację funkcji zarządzających środowiskiem wieloaplikacyjnym na karcie,
- ISO7816-15 - określa sposoby zarządzania obiektami kryptograficznymi na karcie (certyfikaty, klucze) oraz wprowadza jako format opisu pól danych standard ASN.1

2.4.2 Standard GlobalPlatform

GlobalPlatform jest standardem definiującym organizację programistycznych elementów karty w sposób zapewniający jej zgodność z wieloma aplikacjami i rozwiązaniami sprzętowymi. Dodatkowo definiuje on zalecenia dla sprzętu korzystającego z kart (czytniki, bankomaty).

Najważniejszym elementem specyfikacji GP jest *Card Manager* - zarządca karty. Jest to aplet implementujący między innymi podstawowe polecenia ISO7816-4, umożliwiający:

- instalację lub deinstalację aplikacji,
- zarządzanie cyklem życia aplikacji,
- zarządzanie kluczami karty,
- tworzenie **domen bezpieczeństwa** (*Security Domain*),
- obsługa bezpiecznego kanału (*Secure Messaging*).

Każda z ww. operacji wymaga uwierzytelnienia się do karty specjalnym kluczem (lub zestawem kluczy), pozwalającymi na zapewnieniu poufności, integralności i niezaprzeczal-

ności wymienianych informacji.

Domeny bezpieczeństwa to specjalnie wydzielone obszary logiczne, pozwalające na separację grup aplikacji między sobą. Każda domena bezpieczeństwa ma możliwość posiadania indywidualnych zestawów kluczy. Dzięki temu możliwe jest na przykład używanie różnych aplikacji od różnych dostawców w niezależnych od siebie systemach korzystających z różnych kluczy do realizacji bezpiecznej transmisji. Możliwe jest także zarządzanie jedną kartą przez wiele podmiotów bez używania jednego, współdzielonego klucza. Dzięki temu różni dostawcy mają swobodę zarządzania aplikacjami w obrębie swojej domeny bezpieczeństwa, jednocześnie nie posiadając dostępu do obszarów innych producentów.

Standard GlobalPlatform określa także dodatkowe polecenia jakie karta musi implementować aby być w pełni z nim zgodna. W zaprojektowanym aplecie wykorzystywane są komendy związane z zestawianiem bezpiecznego kanału pomiędzy kartą a komputerem, których programistyczna obsługa w pełni została pokryta przez producenta kart będącymi blankietami ELS.

Rozdział 3

Biometria

Niniejszy rozdział omawia podstawy teoretyczne zastosowania cech biometrycznych w informatyce oraz prezentuje praktyczne rozwiązanie problemu uwierzytelnienia użytkownika na podstawie linii papilarnych.

3.1 Model matematyczny cech biometrycznych

Sposób pobierania cech biometrycznych i ich wyjściowy rezultat wymusza na projektantach oprogramowania opracowanie sposobu przetworzenia pobranej cechy na model przyjazny dla algorytmu porównującego. W praktyce stosowane są dwie formy:

- plik graficzny, który przetwarzany jest technikami przetwarzania obrazów pozwalającymi na obliczenie podobieństwa dwóch odcisków palców
- utworzenie specjalnego wzorca zawierającego charakterystyczne cechy pobranej próbki wraz z ich matematyczną reprezentacją, które następnie są porównywane

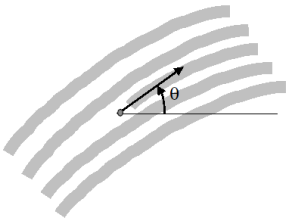
W niniejszej pracy skupiono się na drugiej ze stosowanych postaci, która jest podstawą dla rozwiązania problemu weryfikacji biometrycznej w zaprojektowanym rozwiązaniu.

3.1.1 Matematyczna reprezentacja cech odcisku palca na postawie standardu ISO/IEC 19794-2:2005

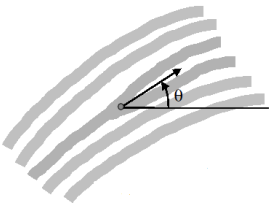
Rozdział opracowano na podstawie dokumentu [30].

Najpopularniejszym wzorcem stosowanym w komercyjnych rozwiązaniach jest ustandaryzowany format zgodny z ISO/IEC 19794-2:2005. Podstawą tego formatu są charakterystyczne punkty występujące na liniach papilarnych zwane *minucjami*. Standard wyróżnia następujące typy minucji:

- zakończenie linii papilarnej (*ridge ending*),
- rozwidlenie (*bifurcation*),
- inne - niezdefiniowane.



Rysunek 3.1: Zakończenie linii papilarnej wraz z zaznaczoną orientacją kątową



Rysunek 3.2: Rozwidlenie linii papilarnej wraz z zaznaczoną orientacją kątową

Format ten przechowywany może być w jednowymiarowej tablicy zmiennych typu *byte* i składa się z czterech składowych:

- nagłówek danych zawierający informacje o rozmiarze danych, wymiarach w pikselach, liczbie odcisków palców oraz inne dane
- nagłówek danych pojedynczego odcisku palca, który zawiera między innymi informacje o liczbie minucji oraz jakości danych
- segment zawierający dane o minucjach
- nieobowiązkowy segment dla danych dodatkowych

Każda minucja opisana jest przez 6 liczb całkowitych (każda przyjmująca wartości z przedziału od 0 do 255). Kodują one odpowiednio:

- rodzaj minucji zakodowany w dwóch najbardziej znaczących bitach pierwszej liczby
- pozycja X minucji tworzona poprzez bitową konkatenację pierwszych dwóch liczb, dwa najbardziej znaczące bity nie są brane pod uwagę
- pozycja Y minucji tworzona poprzez bitową konkatenację liczby trzeciej i czwartej, dwa najbardziej znaczące bity nie są brane pod uwagę
- orientacja kątowa zawarta w piątej liczbie (zakres kąta pełnego został proporcjonalnie przekształcony tak, aby w pełni pokryć go w zakresie zmiennej *byte*)
- jakość minucji

Sposób zapisu minucji w tym standardzie pozwala na wygodne ich użycie w urządzeniach wbudowanych. Format ten jest szeroko stosowany w rozwiązaniach komercyjnych. Stanowi on także podstawę dla autorskich rozwiązań.

3.1.2 Model matematyczny zastosowany w projekcie

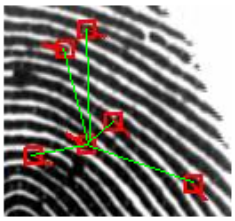
Zbiór minucji znajdujący się w standardowym formacie ISO/IEC 19794-2:2005 nie jest zbiorem uporządkowanym nie daje się porównać bez dokonania pewnych przekształceń.

Ze względu na ograniczone możliwości obliczeniowe karty opracowano wzorzec powstały ze standardowej postaci ISO, który zawiera dane gotowe do porównania.

W nagłówku wzorzec zawiera 3 bajty o stałej wartości (identyfikujące jego format na potrzeby weryfikacji w aplecie) oraz liczbę minucji. Kolejny segment stanowi ciąg informacji o minucjach złożony z dwóch liczb dla każdej z nich:

- typ minucji (zgodnie z formatem ISO/IEC 19794-2:2005),
- orientacja kątowa minucji (zgodnie z formatem ISO/IEC 19794-2:2005)

Trzecia część wzorca zawiera utworzone lokalne struktury, które są porównywane w zaimplementowanym algorytmie. Każda minucja jest łączona z pięcioma najbliższymi sąsiadami. Tworzy to gwiaździstą formę zawierającą 5 par minucji oraz punkt centralny.



Rysunek 3.3: Geometryczna repezentacja porównywalnej struktury lokalnej dla przykładowej minucji

Dla każdej pary obliczane lub wyznaczane są następujące właściwości:

- indeks minucji połączonej z punktem centralnym,
- odległość euklidesowa między minucjami,
- bezwzględna różnica kątowa pomiędzy orientacjami połączonych minucji,
- tangens kąta zawartego między osią OX a odcinkiem utworzonym z pary minucji (zielona linia na rysunku)

Na końcu tak utworzonego wzorca dołączany jest skrót SHA-1 wyliczony ze wszystkich znajdujących się tam danych. Jest on przesyłany do karty wraz ze wzorcem, celem weryfikacji integralności danych odebranych przez aplet.

3.2 Biometria w kontekście kart inteligentnych

3.2.1 Standaryzacja funkcji biometrycznych

Coraz popularniejsze stosowanie biometrii na kartach elektronicznych dało impuls do utworzenia norm standaryzujących proces weryfikacji biometrycznej na tę platformę. Podstawową normą określającą sposób takiego uwierzytelnienia jest dokument ISO7816-11. Określa on format danych biometrycznych przesyłany do karty oraz polecenia, które muszą zostać zaimplementowane dla opisanych wariantów weryfikacji.

Dla kart elektronicznych wspierających platformę JavaCard powstał zestaw interfejsów wspomagających proces projektowania i implementacji własnego rozwiązania biometrycznego. Wydany został on przez organizację JavaCard Forum. Przykładową implementa-

cję apletu oferującego funkcje biometryczne (zarządzanie odciskami palców oraz proces uwierzytelnienia) z użyciem tych bibliotek można znaleźć w [6]. Klasy te są domyślnie zintegrowane z platformą JavaCard w wersji 2.2.2.

3.2.2 Algorytm Match-on-card

Istotną częścią systemów biometrycznych jest algorytm dokonujący oceny podobieństwa dwóch próbek biometrycznych. W systemach wykorzystujących w swojej architekturze karty inteligentne można stosować następujące rozwiązania programowe:

- Match-off-card - karta służy tylko i wyłącznie jako nośnik informacji biometrycznych; dane są z niej pobierane przy każdej próbie weryfikacji, która dokonywana jest na przykład na stacji roboczej
- Match-on-card - karta pełni rolę nośnika oraz posiada algorytm weryfikujący cechy biometryczne

Każde z wymienionych rozwiązań posiada swoje wady i zalety. Główną wadą rozwiązania pierwszego jest zwiększone ryzyko wycieku wzorca biometrycznego. W takim przypadku dana cecha jest na zawsze dyskwalifikowana do użytku w jakimkolwiek innym systemie ze względu na możliwość wykorzystania przechwyconego modelu przez osoby do tego niepowołane. W drugim przypadku wzorzec przechowywany jest w sposób dużo bezpieczniejszy[1], na przeszkodzie w implementacji skutecznego oraz akceptowalnego z punktu użytkownika rozwiązania mogą stanąć jednak ograniczone możliwości obliczeniowe procesorów integrowanych w kartach inteligentnych.

Porównywanie wzorców dzieli się na podstawowe dwie kategorie:

- globalne - wszystkie cechy wektora biometrycznego biorą udział w procesie dopasowania próbek; przykładem może być konstrukcja histogramów na podstawie wszystkich minucji odcisków palców pozwalających wyznaczyć podstawowe przekształcenia przestrzeni 2D (rotację, skalowanie, translację),
- lokalne - w procesie dopasowania biorą udział specjalnie skonstruowane struktury złożone z niewielkiego podzbioru cech wzorca; przykładem może być konstrukcja wielokątów, których wierzchołkami są odpowiednio dobrane minucje; na podstawie oceny podobieństw takich struktur podejmowana jest decyzja o dopasowaniu cech do siebie

Algorytmy mogą uwzględniać podstawowe przekształcenia geometryczne oraz zjawiska powodujące zniekształcenie próbki, które zwiększają trudność problemu porównywania dwóch wzorców. Odpowiednie moduły algorytmu niwelujące efekty tych zniekształceń pozwalają na osiągnięcie wysokiej skuteczności w praktycznym działaniu.

Zaprojektowany w pracy algorytm dokonuje lokalnego porównania struktur opisanych w rozdziale 3.1.2. Wynikiem porównania jest uznanie centralnej minucji struktury przysłanej do karty odpowiednikiem centralnej minucji struktury przechowywanej na karcie, bądź brak takiej decyzji. Dzięki zastosowaniu bezwzględnych wartości w wyliczanych we wzorcu cechach uwzględnia on zarówno rotację jak i translację (przesunięcie w osi X lub

[1] Tak długo, dopóki nie zostanie opracowany skuteczny atak umożliwiający bezpośredni dostęp do pamięci karty.

Y) próbek. Algorytm posiada teoretyczną złożoność O(n^2). Jego szczegółowy opis implementacyjny znaleźć można w rozdziale 4.2.6 a analizę wyników jego skuteczności w rozdziale 4.3.2.

3.2.3 Interfejs programistyczny realizujący funkcje związane z biometrią

Do realizacji projektu zgodnie ze sztuką i ogólnie przyjętymi praktykami inżynierii oprogramowania konieczne było użycie specjalnej klasy, dedykowanej do obsługi zadań związanych z biometrią. W rozdziale 3.1.2 opisano ogólnoprzyjęty interfejs wspomagający programistę w procesie tworzenia apletu z funkcjami biometrycznymi. Ze względu jednak na konieczność zastosowania wcześniejszej wersji platformy JavaCard 2.2.1, która jest kompatybilna ze wszystkimi blankietami ELS[2] opracowano własną klasę integrującą wszystkie niezbędne funkcje na potrzeby projektu. Dzięki temu uzyskano rozwiązanie dużo prostsze w implementacji, znacznym kosztem kompatybilności z innymi gotowymi rozwiązaniami i możliwościami integracji z zewnętrznym oprogramowaniem, stosującym się do wcześniej wymienionych norm. Szczegółowy opis implementacji tego obiektu znajduje się w rozdziale 4.2.5.

[2]Wydawanymi przez Międzyuczelniane Centrum Personalizacji Legitymacji Studenckich Politechniki Poznańskiej

Rozdział 4

Projekt jOSOZ

4.1 Etap projektowy

Etap projektowy pracy wykonany został przy użyciu materiałów dostarczonych od opiekuna pracy. Materiały te pochodziły bezpośrednio od osób technicznie zajmujących się kartami w systemie OSOZ. Celem doprecyzowania niektórych kwestii projektowych, wymienionych zostało kilka maili z przedstawicielami klienta. Ze względów logistycznych nie planowano bezpośrednich spotkań roboczych.

4.1.1 Wymagania funkcjonalne

Projekt składa się z dwóch aplikacji:

- **Aplet JavaCard** wgrywany na karty inteligentne pełniące funkcję Elektronicznej Legitymacji Studenckiej
- **Aplikacja demonstracyjna** udostępniająca prosty graficzny interfejs użytkownika. Z pomocą tej aplikacji realizowana jest dwukierunkowa transmisja z kartą.

Określenie wymagań funkcjonalnych oparte zostało na podstawie opisu funkcjonalności karty OSOZ znajdującym się w artykule [5] oraz korespondencji z klientem.

Funkcje wywoływane z poziomu przygotowanej aplikacji demonstrującej możliwości apletu widoczne dla użytkownika końcowego i operatorów kart to:

- personalizacja apletu,
- sprawdzenie stanu personalizacji apletu,
- sprawdzenie stanu inicjalizacji PINu,
- sprawdzenie stanu inicjalizacji wzorca biometrycznego,
- wgranie wzorca biometrycznego na kartę,
- ustawienie PINu apletu,
- odczyt danych z karty,
- weryfikacja biometryczna,
- weryfikacja PINem,

- sprawdzenie stanu weryfikacji PINem w sesji,
- sprawdzenie stanu weryfikacji biometrycznej w sesji,
- zmiana PINu,
- odblokowanie wzorca biometrycznego,
- generowanie kodu autoryzacyjnego w systemie OSOZ,
- podpisywanie świadczeń za pomocą karty,
- pobranie klucza publicznego karty.

Wszystkie te funkcje realizują niezbędną funkcjonalność, która oferowana jest przez karty wykorzystywane w systemie OSOZ.

4.1.2 Wymagania pozafunkcjonalne

Aby zapewnić odpowiednią jakość oprogramowania i wygodę użytkowania dla końcowych odbiorców, dla obu części składowych pracy określono wymagania pozafunkcjonalne. Wyznaczają one akceptowalne dla posiadaczy kart limity czasowe odpowiedzi na polecenia oraz poziom skomplikowania interfejsu graficznego. Dodatkowo narzucają one wykorzystywanie ogólnoprzyjętych standardów oraz mechanizmów bezpieczeństwa. Dodatkowo określają maksymalne wykorzystanie zasobów karty inteligentnej i poziomy kompatybilności

Dla apletu JavaCard określono następujące wymogi:

- odpowiedź na polecenia w czasie mniejszym niż 1 sekunda,
- wykorzystanie RAMu na poziomie nie większym niż 2kB,
- rozmiar apletu nie większy niż 15 kB,
- czas wykonania algorytmu Match-on-card mniejszy niż 5 sekund,
- mechanizm weryfikacji integralności załadowywanego odcisku palca,
- obsługa PINu zgoda z normą ISO7816-4,
- zgodność uwierzytelniania biometrycznego z normą ISO7816-11,
- mechanizm dostępu do plików uwzględniający dostęp swobodny lub po uprzednim uwierzytelnieniu dowolną metodą,
- możliwość przesyłania danych do karty poprzez bezpieczny kanał,
- generowanie kluczy szyfrujących bezpośrednio na karcie,
- zgodność z bibliotekami JavaCard 2.2.1 oraz GlobalPlatform 2.1.1,
- dedykowany obiekt do obsługi biometrii udostępniający uniwersalne metody, które można wykorzystać w innych projektach.

Dla aplikacji demonstracyjnej założono:

- okienkowy interfejs,
- wykonanie poleceń nie więcej niż po dwóch krokach,
- obsługę czytników zgodnych z interfejsem PC/SC,

- wizualizacja odcisków palcy przechwytywanych z czytnika biometrycznego,
- obsługę blankietów, na których wydawana jest ELS [1].

Wymagania pozafunkcjonalne zostały określone na podstawie pozycji [6],[7],[8],[9].

4.1.3 Przypadki użycia

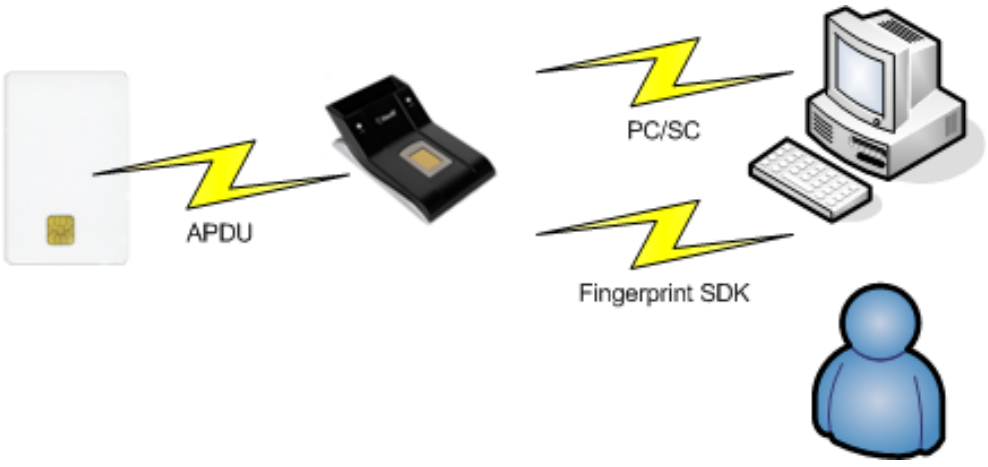
Przypadki użycia można znaleźć w załączniku (*Dodatek A*).

4.1.4 Architektura systemu

Na potrzeby pracy magisterskiej architektura projektu została uproszczona. Przyjęta koncepcja konstrukcji projektu, pomimo swojej prostoty, umożliwia pełne wykorzystanie funkcjonalności napisanego apletu. W skład fizycznej architektury wchodzą:

- komputer klasy PC z systemem operacyjnym obsługującym czytniki w standardzie PC/SC,
- jeden czytnik kart inteligentnych stykowy zgodny ze standardem PC/SC,
- jeden czytnik biometryczny obsługiwany przez oprogramowanie Griaule Fingerprint SDK 2009[2].

Przesyłanie danych między kartą a komputerem następuje poprzez sprzętowe interfejsy komputera (zwykle USB lub COM). Następnie czytnik wysyła komendy APDU bezpośrednio do karty. Informacje zwracane przez kartę czytnik odsyła do komputera. Programową obsługę komunikacji pomiędzy kartą a czytnikiem realizują biblioteki komunikujące się z czytnikiem PC/SC dostępne w wybranym języku programowania.



Rysunek 4.1: Architektura systemu

[1] Modele blankietów, na których personalizowane są ELS to Oberthur ID-One Cosmo V7 oraz Oberthur ID-One Cosmo 64 v5.4

[2] Listę obsługiwanych czytników można znaleźć w [12]

4.1.5 Natywne funkcje apletu

Poniższe tabele opisują zestaw natywnych poleceń apletu jOsoz. Ze względu na ich niestandardowy charakter i brak zdefiniowania w normach standaryzujących przyjęto klasę instrukcji 0xC0, bajty instrukcji liczone są od 0x00 od 0x0A.

Poszczególne elementy tabel opisują:

- **CLA** - bajt klasy instrukcji,
- **INS** - bajt instrukcji,
- **P1** - pierwszy bajt parametru instrukcji,
- **P2** - drugi bajt parametru instrukcji,
- zawartość pola danych,
- wymóg bezpiecznej transmisji danych polecenia,
- **SW** - możliwe statusy wykonania polecenia.

1.Personalizacja apletu

CLA	0xC0
INS	0x00
P1	0x00
P2	0x00
Pole danych	Numer karty w systemie OSOZ w tagu 44 PUK karty w tagu 45
Opis	Komenda personalizuje aplet. Generuje parę kluczy RSA, zapisuje na karcie jej identyfikator w systemie OSOZ, inicjuje obiekt PIN oraz biometryczny.
Wymagany Secure Messaging	Tak
SW	Bezbłędne wykonanie - 9000 Ponowna personalizacja apletu - 6986 Wprowadzone błędne lub niekompletne dane - 6984 Brak Secure Messagingu - 6982

2.Sprawdzenie stanu personalizacji apletu

CLA	0xC0
INS	0x01
P1	0x00
P2	0x00
Pole danych	Niewykorzystywane
Opis	Komenda zwraca stan personalizacji apletu w polu danych odpowiedzi (0001 jeśli aplet jest spersonalizowany, w przeciwnym wypadku 0000)
Wymagany Secure Messaging	Nie
SW	Prawidłowe wykonanie - 9000

3.Sprawdzenie stanu inicjalizacji PINu

CLA	0xC0
INS	0x02
P1	0x00
P2	0x01
Pole danych	Niewykorzystywane
Opis	Komenda zwraca stan inicjalizacji PINu w polu odpowiedzi (0001 jeśli PIN został zainicjowany, w przeciwnym wypadku 0000)
Wymagany Secure Messaging	Nie
SW	Bezbłędne wykonanie - 9000 Aplet niespersonalizowany - 6986 Błędne pole P1 lub P2 - 6A86

4.Sprawdzenie stanu inicjalizacji biometryki

CLA	0xC0
INS	0x03
P1	0x00
P2	0x00
Pole danych	Niewykorzystywane
Opis	Komenda zwraca stan inicjalizacji biometryki w polu odpowiedzi (0001 jeśli odcisk palca został wgrany, w przeciwnym wypadku 0000)
Wymagany Secure Messaging	Nie
SW	Bezbłędne wykonanie - 9000 Aplet niespersonalizowany - 6986 Błędne pole P1 lub P2 - 6A86

5.Pobranie zawartości pliku

CLA	0xC0
INS	0x04
P1	0x00
P2	0xXX XX = 00 - pobranie numeru OSOZ
Pole danych	Niewykorzystywane
Opis	Komenda zwraca w polu danych odpowiedzi dane odczytane z pliku.
Wymagany Secure Messaging	Nie
SW	Bezbłędne wykonanie - 9000 Aplet niespersonalizowany - 6986 Błędne pole P1 lub P2 - 6A86 Wymagane uwierzytelnienie PINem lub biometryczne - 6982

6.Stan uwierzytelnienia PINem w sesji

CLA	0xC0
INS	0x05
P1	0x00
P2	0x01
Pole danych	Niewykorzystywane
Opis	Komenda zwraca stan weryfikacji PINem w sesji
Wymagany Secure Messaging	Nie
SW	Dokonano uwierzytelnienia PINem w sesji - 9000 Brak uwierzytelnienia PINem w sesji - 6300 Błędne pole P1 lub P2 - 6A86 Aplet niespersonalizowany - 6986 PIN nie został zainicjowany - 6985

7.Stan uwierzytelnienia biometrycznego w sesji

CLA	0xC0
INS	0x06
P1	0x00
P2	0x00
Pole danych	Niewykorzystywane
Opis	Komenda zwraca stan weryfikacji PINem w sesji
Wymagany Secure Messaging	Nie
SW	Dokonano uwierzytelnienia biometrycznego w sesji - 9000 Brak uwierzytelnienia biometrycznego w sesji - 6300 Błędne pole P1 lub P2 - 6A86 Aplet niespersonalizowany - 6986 Odcisk palca nie został wgrany - 6985

8.Wygenerowanie kodu OSOZ

CLA	0xC0
INS	0x07
P1	0x00
P2	0x00
Pole danych	Dane zawierająca identyfikator autoryzacji w tagu 41
Opis	Komenda zwraca kod autoryzacyjny OSOZ wraz z jego sygnaturą w polu danych odpowiedzi
Wymagany Secure Messaging	Tak
SW	Prawidłowe wykonanie komendy - 9000 Aplet niespersonalizowany - 6986 Brak Secure Messagingu - 6982 Brak identyfikatora autoryzacji - 6984

9.Podpisanie wiadomości

CLA	0xC0
INS	0x08
P1	0x00
P2	0x00
Pole danych	Dane zawierająca identyfikator autoryzacji w tagu 41 oraz wiadomość do podpisania w tagu 42
Opis	Komenda zwraca wiadomość wraz z jej sygnaturą w polu danych odpowiedzi
Wymagany Secure Messaging	Tak
SW	Prawidłowe wykonanie komendy - 9000 Aplet niespersonalizowany - 6986 Brak Secure Messagingu - 6982 Brak identyfikatora autoryzacji lub wiadomości do podpisania - 6984

10. Pobranie modulusa klucza publicznego karty

CLA	0xC0
INS	0x09
P1	0x00
P2	0x00
Pole danych	Niewykorzystywane
Opis	Komenda zwraca modulus klucza publicznego karty w polu danych odpowiedzi
Wymagany Secure Messaging	Tak
SW	Prawidłowe wykonanie komendy - 9000 Aplet niespersonalizowany - 6986 Brak Secure Messagingu - 6982

11. Pobranie eksponentu klucza publicznego karty

CLA	0xC0
INS	0x0A
P1	0x00
P2	0x00
Pole danych	Niewykorzystywane
Opis	Komenda zwraca eksponent klucza publicznego karty w polu danych odpowiedzi
Wymagany Secure Messaging	Tak
SW	Prawidłowe wykonanie komendy - 9000 Aplet niespersonalizowany - 6986 Brak Secure Messagingu - 6982

4.1.6 Funkcje zgodne z normą ISO7816-4

Poniższe tabele opisują zestaw poleceń zgodnych z normą ISO7816-4. Instrukcje te mają zarezerwowaną klasę 0x00.

1.Inicjalizacja PINu (PUT DATA)

CLA	0x00
INS	0xDB
P1	0x00
P2	0x01
Pole danych	Kod PIN
Opis	Komenda inicjalizuje PIN.
Wymagany Secure Messaging	Tak
SW	Bezbłędne wykonanie - 9000 PIN zainicjowany wcześniej - 6985 Aplet niespersonalizowany - 6986 Niewłaściwy format kodu PIN - 6984 Brak Secure Messagingu - 6982 Błędne pole P1 lub P2 - 6A86

2.Wgranie odcisku palca (PUT DATA)

CLA	0x00
INS	0xDA
P1	0x7F
P2	0x2E
Pole danych	Pierwsze wysłanie polecenia zawiera liczbę bajtów do odebrania Kolejne wywołania polecenia przyjmują przychodzące dane i zapisują je do obiektu obsługującego biometrię.
Opis	Komenda wgrywa odcisk palca na kartę. Dokonuje weryfikacji odcisku z jego haszem SHA1 zawartym w ostatnich dziesięciu bajtach danych.
Wymagany Secure Messaging	Tak
SW	Bezbłędne wykonanie każdego wywołania - 9000 Odcisk palca zainicjowany wcześniej - 6985 Aplet niespersonalizowany - 6986 Niewłaściwy format odcisku palca lub niespójne dane - 6984 Brak Secure Messagingu - 6982 Błędne pole P1 lub P2 - 6A86

3.Odblokowanie odcisku palca (RESET RETRY COUNTER)

CLA	0x00
INS	0x2C
P1	0x00
P2	0x00
Pole danych	Kod PUK zawarty w tagu 45
Opis	Komenda odblokowuje uwierzytelnianie biometryczne po uprzednim zablokowaniu wzorca.
Wymagany Secure Messaging	Tak
SW	Bezbłędne wykonanie każdego wywołania - 9000 Odcisk palca zainicjowany wcześniej - 6985 Aplet niespersonalizowany - 6986 Niewłaściwy format odcisku palca lub niespójne dane - 6984 Brak Secure Messagingu - 6982 Błędne pole P1 lub P2 - 6A86

4.Zmiana PINu (CHANGE REFERENCE DATA)

CLA	0x00
INS	0x24
P1	0x00
P2	0x01
Pole danych	Nowy kod PIN
Opis	Komenda dokonuje zmiany PINu. Wymagane jest uprzednie uwierzytelnienie kodem PIN w sesji.
Wymagany Secure Messaging	Tak
SW	Bezbłędne wykonanie każdego wywołania - 9000 Brak uwierzytelnienia w sesji - 6300 PIN niezainicjowany - 6985 Aplet niespersonalizowany - 6986 Niewłaściwy format odcisku palca lub niespójne dane - 6984 Brak Secure Messagingu - 6982 Błędne pole P1 lub P2 - 6A86

4.Uwierzytelnienie PINem lub biometryczne (VERIFY)

CLA	0x00
INS	0x20
P1	0x00
P2	0xXX XX = 00 - biometryczne XX = 01 - PIN
Pole danych	Kod PIN lub ciąg danych opisany w poleceniu wgrania odcisku na kartę (PUT DATA)
Opis	Komenda dokonuje uwierzytelnienia do apletu podanym sposobem
Wymagany Secure Messaging	Biometrycznie - tak PIN - nie
SW	Bezbłędne wykonanie każdego wywołania - 9000 Błąd uwierzytelnienia - 63CX (XX oznacza liczbę pozostałych prób) PIN lub biometria niezainicjowane - 6985 Aplet niespersonalizowany - 6986 Niewłaściwy format danych uwierzytelniających - 6984 Brak Secure Messagingu - 6982 Błędne pole P1 lub P2 - 6A86

4.1.7 Obsługiwane polecenia standardu GlobalPlatform 2.1.1

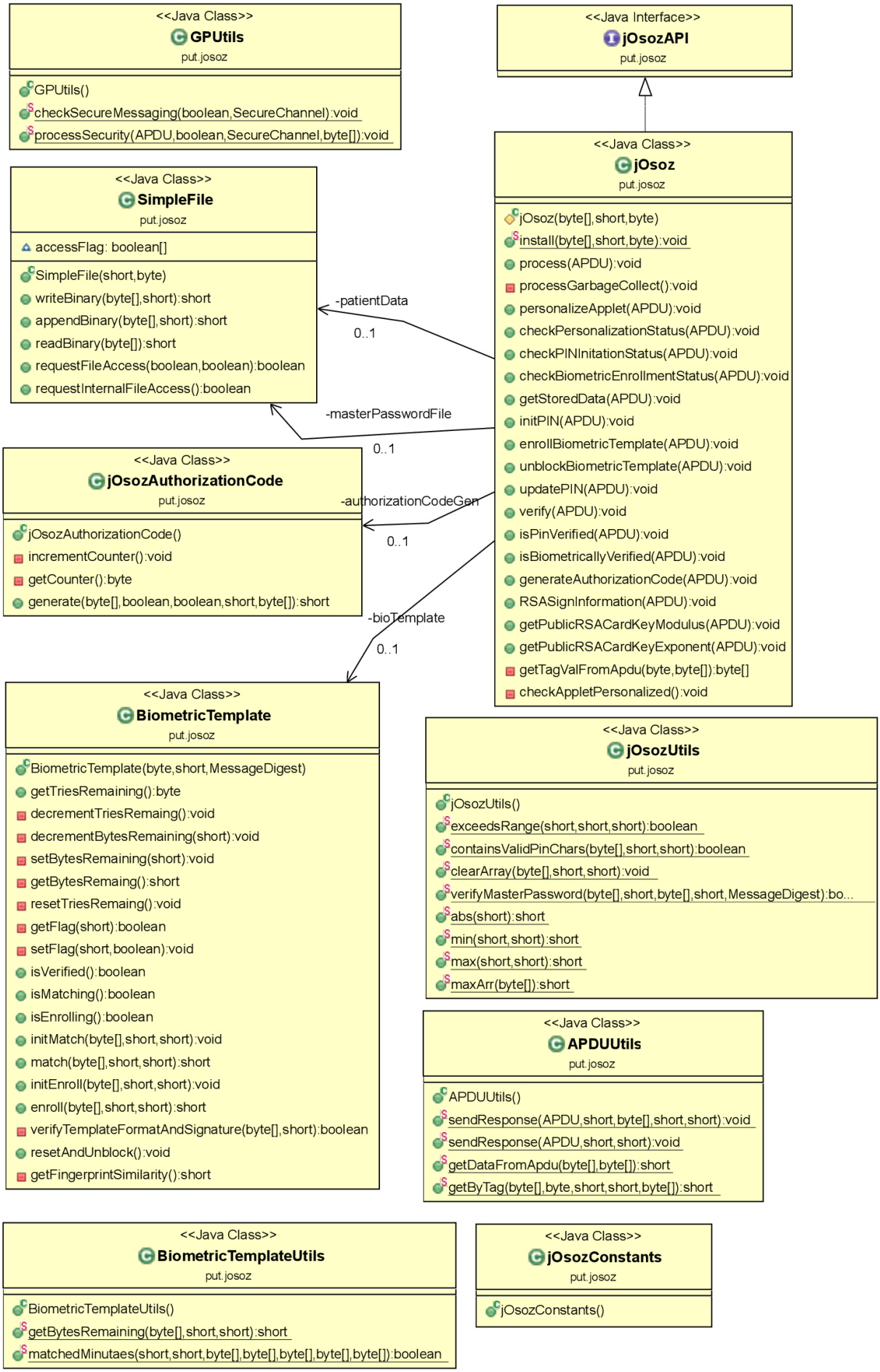
Aplet przechwytuje polecenia z rodziny 0x80 i przekazuje ich wykonanie do modułu karty inteligentnej implementującego polecenia zgodnie ze standardem [13]. To czy polecenie zostanie wykonane w pełni zależy od sposobu implementacji standardu na danym blankiecie posiadającym aplet jOsoz. Aby poprawnie obsługiwać polecenia blankiet musi implementować minimum:

- INIT UPDATE (INS=0x50)
- EXTERNAL AUTHENTICATE (INS=0x82)

4.1.8 Diagram klas apletu oraz ich opis

Aplet został napisany wykorzystując w pełni własności programowania obiektowego[3]. Projekt klas i zależności między nimi prezentuje poniższy diagram.

[3]Język Java wymusza na programiście stosowanie paradygmatu programowania obiektowego niezależne od platformy



Rysunek 4.2: Diagram klas UML

Poszczególne klasy posiadają następujące role:

- jOsozAPI - interfejs, który definiuje wymagane przez aplet metody,
- jOsoz - rozszerzenie klasy Applet zawartej w API JavaCard, implementuje interfejs jOsozAPI. Jest to główna klasa obsługująca wysyłane do apletu polecenia oraz definiująca i wykorzystująca inne obiekty,
- APDUUtils - klasa zawierająca statyczne metody obsługujące pobieranie danych z polecenia APDU,
- BiometricTemplate - klasa obsługująca proces wgrywania odcisku na kartę (*enrollment*) oraz weryfikacji biometrycznej (*Match-on-card*). Posiada też metodę weryfikującą format odcisku palca oraz jego spójność,
- BiometricTemplateUtils - klasa pomocnicza zawierająca metody obliczające podobieństwo porównywanych odcisków oraz metodę parsującą liczbę bajtów do odebrania od komputera,
- GPUtils - klasa pomocnicza posiadająca metody obsługujące bezpieczny kanał,
- SimpleFile - klasa implementująca prosty plik transparentny oraz prawa dostępu,
- jOsozAuthorizationCode - obiekt obsługujący generowanie kodu OSOZ oraz podpisywanie wiadomości,
- jOsozConstants - klasa zawierająca statycznie zdefiniowane stałe do bezpośredniego i wygodnego zastosowania w pisanym kodzie,
- jOsozUtils - klasa zawierająca metody pomocnicze ogólnego przeznaczenia (weryfikacja danych, weryfikacja kodu PUK, proste funkcje matematyczne).

4.1.9 Aplikacja na komputery PC prezentująca możliwości apletu

Szkielet aplikacji desktopowej na komputery oparty został na dostarczonej z SDK przykładowej aplikacji Java korzystającej z czytnika biometrycznego [14]. Formularz aplikacji rozbudowano o akcje związane z funkcjonalnością karty. Dodatkowo dodano klasę konwertującą format ISO/IEC 19794-2:2005 na wzorzec odcisku palca wykorzystywany przez aplet.

4.1.10 Platforma deweloperska

Podstawą platformy deweloperskiej był komputer klasy PC wyposażony w procesor Pentium 4 3.0 GHz, 2GB pamięci RAM. Sterowany był systemem Windows 7 Professional w wersji 64-bitowej. Zintegrowanym środowiskiem deweloperskim wykorzystanym do tworzenia i kompilacji kodu był 64-bitowy Eclipse Kepler, wykorzystujący 64-bitowe biblioteki Java Runtime Environment w wersji 7. Komputer posiadał stykowy czytnik zgodny ze standardem PC/SC SCR 3310 oraz czytnik biometryczny Upek TCS1. Oba urządzenia zintegrowane były w jednym produkcie firmy Zvetco o nazwie Vefifi P5500[15]. Czytnik ten cechuje się wysoką rozdzielczością skanowania (508 DPI), dużą powierzchnią skanowania (12.8mm X 18.0mm). Dodatkowo korzystano także z prostszego urządzenia Zvetco P4000

(czytnik Authentec 4000) o rozdzielczości około 300 DPI bez wbudowanego czytnika kart inteligentnych oraz mniejszym obszarem skanowania [16].

Do kompilacji apletu wykorzystano biblioteki JavaCard 2.2.1, ze względu na zachowanie kompatybilności apletu ze starszymi kartami. Projekt kompilowany był skryptem ANT, będący dziełem autora pracy [4], dostarczony przez Dział Rozwoju Oprogramowania Politechniki Poznańskiej. Do kompilacji wykorzystano także interfejsy napisane w języku Java dla biblioteki GlobalPlatform 2.1.1, które również zostały dostarczone przez DRO. Budowanie projektu odbywało się w trybie zgodności klas Java w wersji 1.4. Do kompilacji aplikacji desktopowej wykorzystano produkt Griaule Fingeprint SDK 2009, dostarczony z zakupionymi przez Politechnikę Poznańska czytnikami biometrycznymi. Dodatkowo wykorzystano biblioteki realizujące obsługę bezpiecznego kanału z projektu GPJ [17], które dostosowano na potrzeby pracy. Wgrywanie apletu oraz początkowe testy realizowane były z pomocą otwartoźródłowego programu GPShell [18].

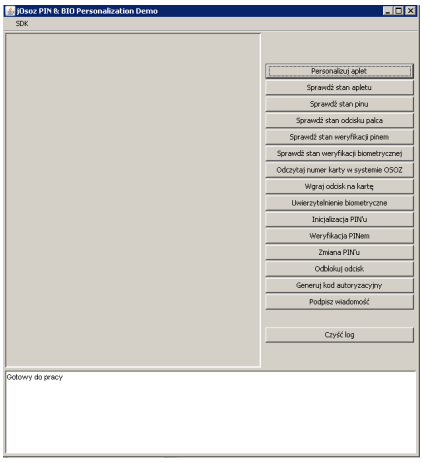
4.2 Implementacja

4.2.1 Realizacja systemu plików

Pliki obsługiwane są przez klasę SimpleFile. Jest to klasa udostępniająca proste metody odczytu całości, zapisu oraz dopisania do końca pliku. Każde polecenie zapisu do pliku powoduje nadpisanie aktualnej jego zawartości. Możliwe jest określenie maksymalnej pojemności pliku. Sama zawartość przechowywana jest w prostej tablicy bajtów. Dostęp do plików nie jest zgodny z API określonym w normie ISO7816-4, jednak implementacja według tego standardu nie wymagałaby dużego nakładu pracy przy zachowaniu aktualnej funkcjonalności klasy.

Aplet w chwili obecnej posiada 2 pliki:

- plik zawierający numer karty w systemie OSOZ. Jest to plik, do którego dostęp do odczytu jest swobodny (nie jest wymagane uwierzytelnienie PINem ani biometryczne). Dostęp można uzyskać poprzez polecenie natywne odczytu pliku z para-



Rysunek 4.3: Interfejs aplikacji prezentującej możliwości apletu

metrem P2=00,

- plik zawierający skrót kodu PUK karty uzyskany poprzez zakodowanie go algorytmem SHA1. Dostęp do tego pliku możliwy jest tylko wewnątrz apletu.

4.2.2 Mechanizm dostępu do plików

Klasa SimpleFile umożliwia nadanie uprawnień dostępów do odczytu w momencie tworzenia plików, poprzez umieszczenie odpowiednich reguł dostępu w konstruktorze obiektu. Dostęp realizowany jest poprzez złożenie żądania dostępu do zawartości pliku i jego wyniku do specjalnej flagi. Przy wykonywaniu odczytu jej zawartość jest sprawdzana i konsumowana (poprzez ustawienie wartości *false*). Następnie uzyskiwany jest dostęp do zawartości pliku. Flaga ważna jest do momentu odłączenia karty od zasilania[4] lub do momentu wykonania odczytu pliku. Do składania żądań dostępu do plików służą 2 metody:

- requestFileAccess(bool, bool) - przyjmującą w pierwszym parametrze wynik uwierzytelnienia PINem, w drugim wynik uwierzytelnienia biometrycznego,
- requestInternalFileAccess() - wykorzystywana do dostępu do plików wykorzystywanych wewnętrznie (np PUK karty),

Możliwe jest określenie następujących poziomów dostępu do odczytu plików:

- dostęp swobodny (0x00) - do odczytu zawartości nie jest wymagane uwierzytelnienie kodem PIN lub poprzez odcisk palca,
- dostęp po uwierzytelnieniu (0x01) - wymagane jest uwierzytelnienie biometryczne lub poprzez PIN,
- dostęp wewnętrzny (0x02) - dostęp tylko i wyłącznie po wywołaniu metody requestInternalFileAccess.

Dostęp do zapisu lub dopisania zawartości do każdego pliku został programistycznie zabezpieczony poprzez wymaganie zestawionego bezpiecznego kanału[5].

4.2.3 Obsługa PIN-u

Obsługa PINu została oparta na wbudowanej w JavaCard klasie OwnerPIN. Udostępnia ona zestaw metod, które w pełni spełniają założenia projektu odnośnie obsługi PIN. Dodatkowo klasa ta, wspiera sprawdzenie stanu uwierzytelnienia w sesji[6].

Docelowo, oprócz weryfikacji z wykorzystaniem komputera, uwierzytelnienie PINem do systemu OSOZ może odbyć się poprzez urządzenie typu PINPAD. Z tego powodu obsługa weryfikacji została zaprojektowana w pełnej zgodności z normą ISO7816-4. Ze względu na niedostępność takich urządzeń funkcja ta nie została jednak na nich przetestowana.

[4]Flaga przechowywana jest w nietrwałej pamięci czyszczonej po resecie karty (CLEAR_ON_RESET).

[5]Zestawienie bezpiecznego kanału wymaga znajomości specjalnego klucza karty, do którego dostęp z założenia mają nieliczne osoby.

[6]W klasie tej użyto nietrwałej pamięci typu CLEAR_ON_RESET. Przy użyciu tego typu obszatu pamięci sesja trwa do momentu resetu karty.

4.2.4 Wykorzystane funkcje i obiekty kryptograficzne

Aby zaimplementować funkcje związane z podpisywaniem wiadomości wykorzystano zawarte w JavaCard klasy:

- KeyPair - obiekt generujący parę kluczy o wcześniej ustalonym typie. Udostępnia metody pozwalające na pobranie wygenerowanych kluczy,
- RSAPrivateKey - dedykowany obiekt do bezpiecznego przechowywania klucza prywatnego karty,
- RSAPublicKey - dedykowany obiekt do bezpiecznego przechowywania klucza publicznego karty,
- RSASigner - obiekt udostępniający mechanizmy podpisu kluczem prywatnym oraz jego weryfikacji kluczem publicznym.

Wykorzystanie ww. obiektów jest jedyną zalecaną metodą. Zastosowane w pracy zalecenia projektowe [9] zdecydowanie odradzają przechowywanie kluczy w prostych tablicach ze względów bezpieczeństwa. Użycie wspomnianych klas powoduje także wykorzystanie wbudowanego procesora kryptograficznego, który pozwala wykonać wszelkie operacje związane z kryptografią w czasie akceptowalnym dla użytkownika[7].

Dodatkowo, aby bezpiecznie przechowywać PUK karty w postaci niejawnej i weryfikować spójność odcisków palców, wykorzystano klasę MessageDigest. Klasa ta udostępnia kilka popularnych funkcji skrótu, w tym szeroko stosowany SHA1. Został on użyty ze względu na jego implementację w wielu innych językach programowania, co w przyszłości ułatwi integrację funkcji apletu z innymi systemami.

4.2.5 Implementacja klasy obiektu biometrycznego

Biblioteki JavaCard w wersji 2.2.2 posiadają wbudowane klasy zapewniające uspójniony interfejs dla programistów dodających funkcje biometryczne do kart inteligentnych. Standardowe wykorzystanie obiektów z tego pakietu zakłada utworzenie dwóch apletów - apletu zarządzającego biometrią na karcie (realizującego procesy związane bezpośrednio z wgrywaniem i obsługą cech biometrycznych znajdujących się na karcie) oraz apletu sprawdzającego podobieństwo cech biometrycznych i zwracającego wynik porównania. Współpraca między tymi apletami odbywa się wtedy za pomocą wbudowanego w JavaCard interfejsu Shareable, który pozwala na interakcje między dwoma apletami załadowanymi na karcie. Szerszy opis wraz z przykładem wykorzystania tych klas można znaleźć w [6],[19].

Ze względu na konieczność zachowania kompatybilności apletu ze starymi modelami blankietów obsługujących biblioteki JavaCard 2.2.1 zaprojektowano i zaimplementowano własną klasę obsługującą biometrię. Nosi ona nazwę BiometricTemplate.

Odchodzi ona od wzorca wspomnianego wcześniej. Zintegrowano w niej obie funkcjonalności (zarządzanie cechami biometrycznymi oraz ich porównywaniem) i umieszczono w jednym aplecie bez wykorzystywania interfejsu typu Shareable. Dla uproszczenia klasa

[7] W praktyce niemożliwym jest oprogramowanie własnej nietrywialnej funkcji kryptograficznej z wykorzystaniem API JavaCard, ze względu na niemożność uzyskania dostępu do koprocesora kryptograficznego z poziomu kodu.

obsługuje jeden wzorzec biometryczny, wgrywany tylko raz w cyklu życia apletu. W przypadku zablokowania wzorca istnieje możliwość jego odblokowania. W ramach klasy zawarty został też algorytm Match-on-card, który szerzej został opisany w rozdziale trzecim, a jego techniczne aspekty zawarte zostały w następnym podrozdziale.

Interfejs klasy inspirowany był implementacją obiektu OwnerPIN API JavaCard i udostępnia następujące metody:

- getTriesRemaining() - metoda zwracająca liczbę prób uwierzytelnienia pozostałych do zablokowania odcisku palca,
- decrementTriesRemaing() - metoda zmniejszająca liczbę prób uwierzytelnienia. Wywoływana przed każdym porównywaniem przysłanego odcisku palca ze wzorcem na karcie, o ile wzorzec nie jest zablokowany,
- decrementBytesRemaining(short value) - metoda zmniejszająca liczbę oczekiwanych bajtów do odebrania przez obiekt,
- setBytesRemaining(short value) - metoda ustawia liczbę oczekiwanych bajtów do odebrania,
- getBytesRemaing() - metoda zwraca aktualną liczbę oczekiwanych bajtów do odebrania,
- resetTriesRemaing() - metoda przywraca licznik prób uwierzytelnienia do maksymalnej, wcześniej ustalonej, wartości; wywoływana po każdej udanej próbie uwierzytelnienia,
- getFlag(short index) - metoda pobiera wybraną flagę sesyjną z tablicy zmiennych określających stan obiektu w sesji z kartą,
- setFlag(short index, boolean value) - metoda ustawia wybraną flagę sesyjną,
- isVerified() - zwraca flagę sesyjną, określającą fakt udanego uwierzytelnienia biometrycznego w sesji,
- isMatching() - metoda zwraca flagę sesyjną, określającą fakt odbywania się procesu porównywania odcisków palcy,
- isEnrolling() - metoda zwraca flagę sesyjną, określającą fakt wgrywania wzorca biometrycznego na kartę,
- initMatch(byte[] matchData, short offset, short length) - metoda odbiera liczbę bajtów nadchodzącego wzorca i przestawia flagę określającą stan porównywania odcisków palcy na *true*,
- match(byte[] incomingData, short offset, short length) - metoda odbiera dane porównywanego odcisku palca; w przypadku odebrania wszystkich danych dokonuje sprawdzenia spójności oraz formatu odcisku, porównania go ze wzorcem przechowywanym na karcie i zwrócenia wyniku uwierzytelnienia,
- initEnroll(byte[] enrollData, short offset, short length) - analogiczna do metody *initMatch*, jednakże stosowana jest tylko i wyłącznie w procesie wgrywania wzorca na kartę,
- enroll(byte[] matchData, short offset, short length) - metoda analogiczna do metody

match; po odebraniu wszystkich danych dokonuje weryfikacji formatu i spójności wzorca i zapisuje go bądź odrzuca,

- verifyTemplateFormatAndSignature(byte[] input, short currentLength) - metoda porównuje skrót SHA1 z odcisku palca z sygnaturą przechowywaną w jego ostatnich 10 bajtach; dodatkowo sprawdza nagłówek (pierwsze 3 bajty) oraz liczbę zawartych punktów charakterystycznych (minucji),
- resetAndUnblock() - metoda ustawiająca z powrotem maksymalną liczbę prób uwierzytelniania na maksymalną i resetująca stan flagi określającej prawidłową weryfikację na *false*; wywoływana przy odblokowywaniu odcisku palca; metoda ta może znaleźć swoje zastosowanie w przypadku umożliwienia zmiany wzorca przechowywanego na karcie,
- getFingerprintSimilarity() - metoda zwraca podobieństwo między dwoma porównywanymi odciskami palców.

W projekcie obiekt zezwala na 10 nieudanych prób uwierzytelnienia do momentu zablokowania wzorca. Dodatkowo można w nim przechować informacje dla maksymalnie 60 minucji pobranych z odcisku palca[8].

Dane o odcisku palca oraz o odcisku przysyłanym celem uwierzytelnienia biometrycznego przechowywane są w pamięci trwałej. Rozwiązanie takie przyjęto ze względów możliwości sprzętowych karty (brak dostępnej nietrwałej pamięci operacyjnej). Flagi sesyjne przechowywane są w pamięci nietrwałej (*transient*) i spełniają następujące funkcje:

- FLAG_VERIFIED - określa, czy nastąpiło udane uwierzytelnienie biometryczne w sesji,
- FLAG_ENROLLING - określa, czy obiekt dokonuje wgrywania odcisku na kartę; pozwala to w trakcie wykonywania serii poleceń przesyłających dane do karty przełączyć się z metody initEnroll do metody enroll,
- FLAG_MATCHING - określa, czy obiekt dokonuje biometrycznego uwierzytelnienia; pełni funkcję analogiczną do flagi FLAG_ENROLLING pozwala przełączyć się z metody initMatch do metody match.

W wypadku odłączenia zasilania karty flagi te są resetowane. W przypadku przerwania procesu porównywania lub wgrywania wzorca na kartę pozwala to obiektowi na przywrócenie stanu sprzed wykonania polecenia i umożliwienie ponownego jego wykonania. W czasie aktywnej sesji flagi te pomagają odpowiednio sterować zachowaniem obiektu na wydawane przez użytkownika polecenia.

Ciekawym problemem jaki został rozwiązany w powyższej klasie była implementacja licznika nieudanych prób. W pozycjach opisano technikę ataku *rollback attack* i wynikające z jego stosowania zagrożenia dla systemu. W uproszczeniu polega on na wykorzystaniu faktu wywołania kodu zmieniającego stan licznika w trakcie transakcji. Mechanizm transakcji w JavaCard pozwala na uzyskanie własności ACID w przypadku zapisywania danych np. do tablic bajtów lub trwałych zmiennych. W przypadku utraty zasilania stan licznika zostanie przywrócony do ostatniego spójnego stanu - przed zwiększeniem. Pozwala to ata-

[8]Typowy zakres liczbowy punktów charakterystycznych pobieranych przez czytnik Zvetco P5500 wynosi od 20 do około 40 minucji.

kującemu na ominięcie blokad wynikających z przekroczenia licznika na karcie i skuteczne atakowanie np. kodu PIN metodą *brute-force*.

Akceptowalnym rozwiązaniem było przeniesienie aktualnego stanu licznika do pamięci nietrwałej i jego zwiększenie poza obszarem objętym transakcją (pamięć nietrwała nie bierze udziału w transakcji [20]). W ten sposób w przypadku ataku na licznik, nie można przewidzieć dokładnej jego wartości po odzyskaniu przez kartę zasilania (według [20] wartość jest teoretycznie losowa, co jest akceptowalne przez standardy określone przez Oracle, taką ochronę przed atakiem stosuje klasa OwnerPIN). Opisaną problematykę wraz z innymi metodami ochrony poruszono w [21].

4.2.6 Algorytm Match-on-card

Algorytm oparty jest na prostych operacjach jakie udostępnia JavaCard. Jego centrum stanowią dwie pętle *for*, iterujące po minucjach wzorca zapisanego na karcie oraz wzorca porównywanego. Ze względu na bardzo ograniczone możliwości obliczeniowe, operacje wymagające funkcji trygonometrycznych lub pierwiastków oraz modelowanie par minucji są realizowane na komputerze i następnie przesyłane w formacie przyjaznym dla środowiska JavaCard. Celem poprawienia wydajności oraz ominięcia niepotrzebnych porównań cech już dopasowanych zastosowano jedną tablicę (mieszczącą się w pamięci RAM) o rozmiarze równym liczbie minucji w porównywanym wzorcu, która mapuje dopasowane już cechy i pozwala na ich pominięcie w wewnętrznej pętli. Algorytm nie korzysta z mechanizmów odcięć ani prostych heurystyk, aby nie utracić dokładności.

Ocena stopnia podobieństwa sąsiedztw obu minucji wykonywana jest za pomocą prostych operacji matematycznych (dodawanie, odejmowanie, mnożenie) i wykorzystanie własnych implementacji funkcji *abs*, *min*, *max*. Dużą część instrukcji stanowi adresowanie odpowiednich wartości, ze względu na zastosowany model matematyczny odcisku palca i przechowywanie go w prostej tablicy bajtów.

Różnice względne między cechami porównywanych par minucji odnoszone są do ustalonych w algorytmie progów. Przyznawane są punkty za zmieszczenie się w ustalonej granicy tolerancji (5 lub 10 w zależności od cechy). Nieznaczne przekroczenie wartości tolerancji skutkuje brakiem przyznania punktów. Znaczna odchyłka jednej cechy dyskwalifikuje parę. Parę uznajemy za dopasowaną, gdy zdobędzie ona przynajmniej 15 punktów. Uzyskanie 25 punktów uważane jest za idealne dopasowanie i promuje sąsiedztwo poprzez podwójne liczenie danej pary. Dwie minucje są uznawane za odpowiadające sobie, gdy przynajmniej trzy z pięciu par zdefiniowanych w ich sąsiedztwach zostały dopasowane do siebie. Oznacza to minimalnie trzy dopasowania par, które zdobyły od 15 do 24 punktów lub dwa dopasowania, z których jedno uzyskało powyżej 25.

Dopasowanie jednej minucji do drugiej zwiększa licznik zmiennej *matched*, która zlicza udane dopasowania. Na koniec zwracany jest wynik w postaci procentowej z równania zaprezentowanego w pracy [22], cytowanego przez wiele innych artykułów poruszających tematykę porównywania ocisków palcy na podstawie minucji:

$$score = \frac{2*matched*100}{n+m}$$

Gdzie m,n to liczba minucji wzorca odpowiednio:

- m - na karcie.

- n - porównywanego.

Przyjęto, że uwierzytelnienie biometryczne zachodzi pomyślnie gdy:

$$score \geq 35$$

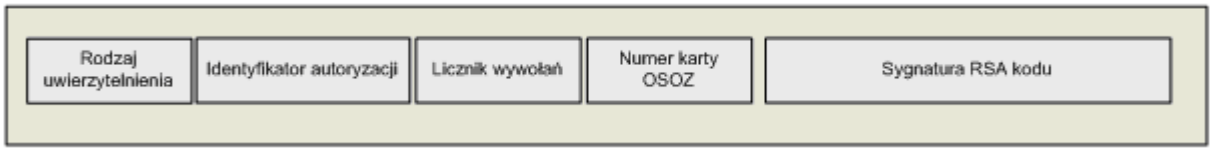
W przeciwnym wypadku karta odsyła odpowiedź zawierającą liczbę prób, które pozostały do zablokowania wzorca.

Pomimo stosunkowo niskiej złożoności obliczeniowej (n^2) algorytm wykonuje się długo (więcej niż 30 s. na nowszych blankietach), co jest nieakceptowalne z punktu widzenia użytkownika i operatora karty. Dalsze prace nad algorytmem skupione będą głównie na kwestii wydajnościowej (próba uzyskania złożoności obliczeniowej zbliżonej do liniowej, stosowanie odcięć lub prostych heurystyk, stosowanie dodatkowych tablic).

4.2.7 Kod OSOZ i podpisywanie wiadomości

Generowanie kodów autoryzacyjnych i podpisywanie świadczeń zostało dostatecznie szczegółowo opisane w [5], by móc zaimplementować tę funkcjonalność. Zaprojektowano specjalny obiekt jOsozAuthorizationCode, który realizuje:

- sprawdzenie rodzaju uwierzytelnienia (biometria, PIN, jedno i drugie) i dodanie go do nagłówka kodu
- przyjęcie danych identyfikujących autoryzację (w pracy wykorzystana została data i godzina przysyłana z komputera[9])
- zliczanie liczby wywołań generacji kodu w sesji i dołączanie jej do nagłówka
- dołączenie do nagłówka kodu numeru karty w systemie OSOZ (generowanie kodu autoryzującego) lub podpisywanej wiadomości[10] (w przypadku chęci podpisywania świadczeń)
- generowanie podpisu RSA (wyznaczenie skrótu SHA1 wiadomości, zastosowanie paddingu PKCS#5 i obliczenie sygnatury) kluczem prywatnym karty nagłówka kodu i dołączenie podpisu do odpowiedzi



Rysunek 4.4: Budowa kodu autoryzacyjnego w systemie OSOZ

Wygenerowany kod wraz z podpisem zwracany jest w pojedynczej odpowiedzi APDU. Jest to możliwe przy zastosowaniu pary kluczy o długości 1024 bitów. Zastosowanie tej długości zawsze prowadzi do wygenerowania 128 bajtów danych sygnatury RSA. Dodatkowo aplet ogranicza długość danych identyfikatora autoryzacji, numeru karty OSOZ oraz

[9] W chwili pisania pracy autor nie uzyskał szczegółów dotyczących budowy identyfikatora autoryzacji w trybie on-line oraz off-line.

[10] W pracy przyjęto, że świadczeniem jest dowolny ciąg bajtów spełniający wymogi co do minimalnej i maksymalnej długości przyjmowanej przez aplet. W chwili pisania pracy autor nie uzyskał informacji o formacie danych wysyłanych do karty w tym trybie.

podpisywanego świadczenia tak, aby całkowita długość danych do zwrócenia była mniejsza bądź równa 256 bajtów.

Aplet udostępnia metody pozwalające na pobranie modulusa i eksponentu klucza publicznego karty, które można łatwo przesłać w polu odpowiedzi APDU. Dzięki temu możliwe jest jego skonstruowanie w urządzeniu zewnętrznym i wykorzystanie do weryfikacji otrzymanych danych.

4.3 Testy

4.3.1 Testy poprawności implementacji funkcji apletu

W każdym projekcie programistycznym pewna część czasu powinna zostać zagospodarowana na testy wytworzonego oprogramowania. W przypadku aplikacji demonstracyjnej przeprowadzono wielokrotne testy funkcjonalne poprzez interakcję użytkownika z graficznym interfejsem. Dodatkowo stosowany był wbudowany w środowisko Eclipse debugger, który skutecznie spełniał swoje zadanie jako odpowiednik testów funkcjonalnych, pozwalając na śledzenie wykonania programu i tym samym wykrywanie błędów na poziomie pojedynczych linii kodu.

Testy przeprowadzono także dla apletu. Pomimo specyficznej architektury wygodne testowanie apletu napisanego w JavaCard może odbywać się w dwojaki sposób:

- na symulatorze (cref) wbudowanym w biblioteki JavaCard. Takie rozwiązanie pozwala na wygodne debugowanie kodu w zintegrowanym środowisku deweloperskim i dokładną weryfikację poprawności działania wytworzonego kodu.
- na blankiecie ELS, z wykorzystaniem oprogramowania przyjmującego skrypty testowe.

Ze względu na niewspieranie przez emulator standardu Global Platform 2.1.1 i tym samym niemożność zestawienia bezpiecznego kanału do weryfikacji poprawności działania kodu zastosowano metodę drugą. Wykorzystano do tego darmowe oprogramowanie GPShell. Pozwala ono na wsadowe testowanie karty przy pomocy wcześniej spreparowanych skryptów testowych, zawierających polecenia APDU. Oprogramowanie to obsługuje przesyłanie poleceń poprzez bezpieczny kanał, co pozwoliło przetestować wszystkie funkcjonalności apletu, bez wyłączania tej funkcji. Stosowne skrypty testowe zostały przygotowane.

Testy finalnej wersji apletu zakończyły się powodzeniem. Nieprawidłowe wywoływanie poleceń kończyło się zawsze zwróceniem wyjątku z odpowiednim kodem odpowiedzi. Prawidłowe przetworzenie polecenia zwracało zawsze oczekiwaną (najczęściej 0x9000) odpowiedź. Żaden z przypadków testowych nie spowodował wystąpienia nieznanego lub nieobsłużonego wyjątku identyfikującego się słowem statusowym 0x6F00.

4.3.2 Skuteczność algorytmu weryfikującego odcisk palca

Testy skuteczności algorytmu, ze względu na ograniczone możliwości czasowe, nie zostały wykonane na kartach z użyciem własnej bazy danych odcisków. Skorzystano z weryfikatora autorstwa Laboratorium Systemów Biometrycznych Uniwersytetu w Bolonii, organi-

zującego zawody Finger Verification Competition [23]. Strona projektu znajduje się pod adresem https://biolab.csr.unibo.it/.

Algorytm został przetestowany przez standardowy tester programów przyjmujący odciski w formacie ISO/IEC 19794-2:2005. Dokonuje on 27720 prób weryfikacji, które powinny zakończyć się powodzeniem oraz 87990 prób, które powinny zakończyć się odmową autoryzacji. Odciski palców w bazie danych pobrane zostały przez sensor optyczny o rozdzielczości 500 dpi. Każdy z nich jest próbką wysokiej jakości pobraną w typowych warunkach dla komercyjnych systemów korzystających z biometrii. Wyniki tego testu z dużym prawdopodobieństwie odzwierciedlają skuteczność algorytmu w środowisku produkcyjnym.

Po wykonaniu testu wyznaczane są następujące miary:

- EER *equal error rate* - średnia liczba błędów pozytywnych algorytmu, dla tego pojęcia obowiązuje także skrót FNMR (*false non match rate*),
- FMR100 *false match rate* - wyznacza najniższe FNMR, dla którego na 100 nieautoryzowanych prób dostępu jedna zakończy się sukcesem,
- FMR1000 - wyznacza najniższe FNMR, dla którego na 1000 nieautoryzowanych prób dostępu jedna zakończy się sukcesem,
- FMR10000 - wyznacza najniższe FNMR, dla którego na 10000 nieautoryzowanych prób dostępu jedna zakończy się sukcesem,
- ZeroFMR - wyznacza najniższe FNMR, dla którego żadna z nieautoryzowanych prób dostępu nie zakończy się sukcesem,
- ZeroFNMR - wyznacza najniższe FMR, dla którego żadna prawidłowa autoryzacja odciskiem palca nie zostanie odrzucona,
- średni czas wykonania algorytmu,
- rozkład wyników pomiarów podobieństw pomiędzy odciskami palców dla obu rodzajów testów,
- wykresy FMR(t) oraz FNMR(t), gdzie t jest granicznym podobieństwem, dla którego 2 odciski uznaje się za sobie odpowiadające.

Mechanizm sprawdzania algorytmu jest analogiczny dla mechanizmów znanych chociażby z wielu edycji Olimpiady Informatycznej. Aby program został poprawnie zweryfikowany musi spełniać następujące wymogi:

- być 32-bitową aplikacją konsolową na platformę Windows,
- przyjmować 3 parametry z linii poleceń (plik z informacjami o odcisku pierwszym, plik z informacjami o odcisku drugim oraz ścieżkę z docelowym plikiem wynikowym),
- musi dopisywać (co wymusza tryb *append* dostępu do pliku) do końca pliku wyjściowego linię zawierającą nazwy porównywanych plików, rezultat przeprowadzenia testu (OK lub FAIL) oraz wynik podobieństwa będącego liczbą zmiennoprzecinkową z zakresu od 0 do 1,
- wysyłany plik wykonywalny musi posiadać nazwę *match.exe* i być spakowanym w archiwum w formacie ZIP,
- wysyłanie programu do testowego sprawdzenia może odbywać się raz na 12 godzin,

- wysyłanie programu do przeprowadzenia pełnego testu może odbyć się raz na 30 dni,
- maksymalny czas wykonania algorytmu - 3 sekundy.

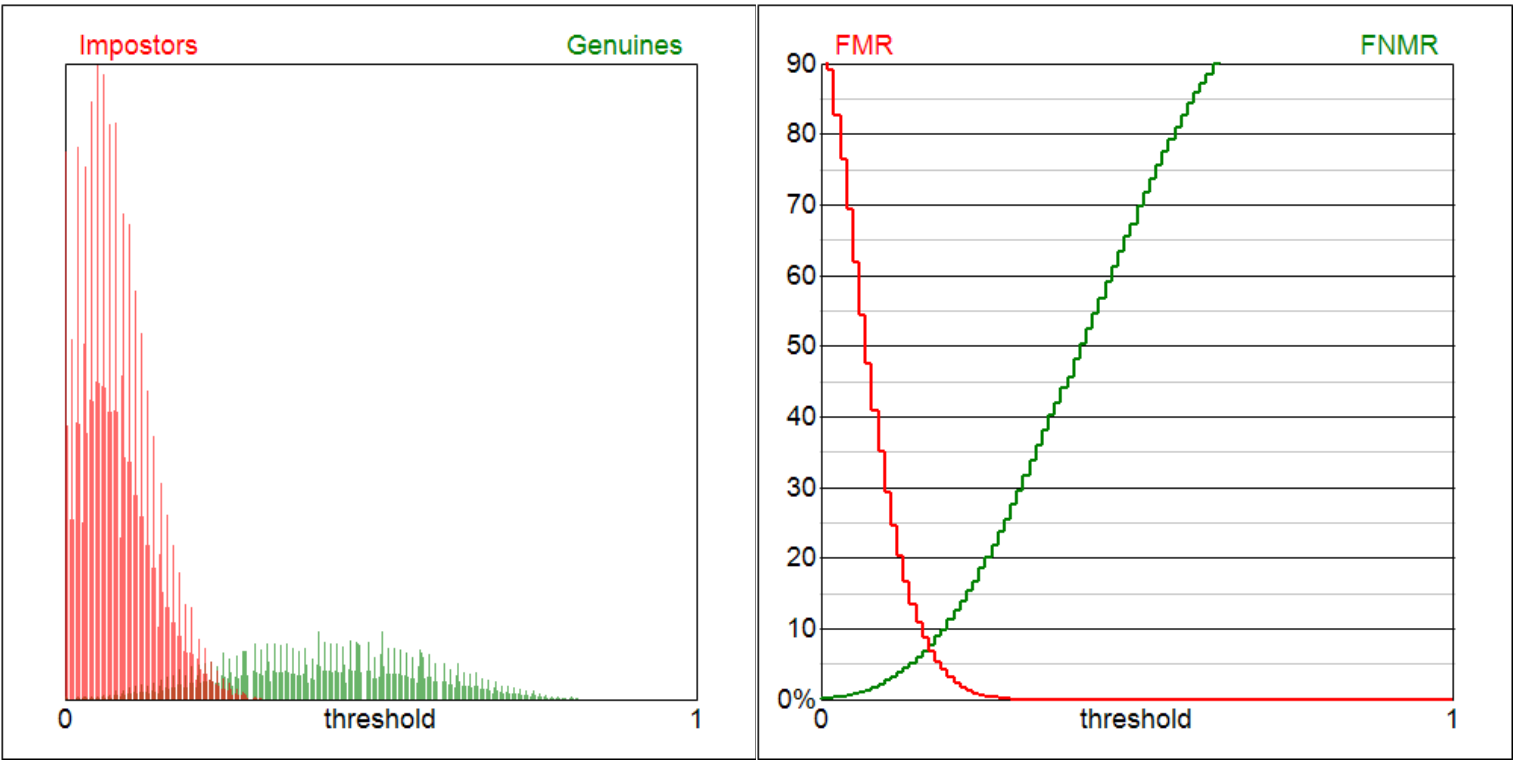
Wyniki testów mogą zostać opublikowane na stronie organizatora. Spowoduje to odpowiednie uwzględnienie ich w rankingu wśród innych opublikowanych wyników algorytmów.

Aby przetestować opracowany algorytm, autor przepisał jego implementację z języków Java (konwersja formatu) i JavaCard (porównywanie) do języka C#. Organizator konkursu udostępnia szkielet programu w języku C++, zawierający klasę do obsługi formatu wykorzystywanego do testów oraz przykładowe pliki wejściowe. Ze względu na problemy z integracją szkieletu z algorytmem autor nie skorzystał z udostępnionych plików.

Wyniki algorytmu prezentują się następująco:

- EER 7,340%,
- FMR100 16,804%,
- FMR1000 27,745%,
- FMR10000 36,017%,
- ZeroFMR 44,040%,
- ZeroFNMR 100%,
- średni czas wykonania algorytmu 51ms.

Wykresy skuteczności prezentują się następująco:



Rysunek 4.5: Wykres dystrybucji wyników (po lewej) oraz wykres zależności FMR i EER od przyjętej wartości granicznej podobieństwa *t*.

Wyniki ukazują, że algorytm może spełniać swoją rolę w środowisku komercyjnym,

jednakże przy założeniu odrzucania co trzeciej lub nawet co drugiej próby pozytywnego uwierzytelnienia się. W praktyce każdy produkt komercyjny oferuje praktycznie zerową wartość FMR przy EER wynoszącym około 1% [24]. Wykresy bardzo wyraźnie pokazują, że skupić się należy przede wszystkim na eliminowaniu błędów pozytywnych (przesunięcie dystrybucji wyników, które powinny zostać uznane za prawidłowe w kierunku wyższych wartości).

4.4 Bezpieczeństwo proponowanego rozwiązania

Dane biometryczne przechowywane w aplecie, w zależności od wielu czynników, mogą być traktowane jako osobowe, które podlegają ochronie zgodnie z ustawą z dnia 29 sierpnia 1997 r. o ochronie danych osobowych [25]. Ustawowym obowiązkiem każdego administratora systemu jest ich odpowiednie zabezpieczenie, przy czym ustawa nie określa jednoznacznie sposobu zabezpieczenia.

Mechanizmy bezpieczeństwa wszelkich danych polegają przede wszystkim na hermetyczności środowiska JavaCard. Z założenia, żadne wrażliwe dane biometryczne nie są przesyłane na zewnątrz karty. Ewentualne dane przechowywane w plikach, które posiadają możliwość odczytu mogą zostać zabezpieczone wymogiem uwierzytelnienia się jedną z dostępnych w aplecie metod.

Bez przeprowadzenia udanego ataku na sam system JavaCard (np. poprzez *buffer overflow*) nie jest możliwym bezpośredni dostęp do pamięci trwałej karty, i tym samym brak jest jakiejkolwiek możliwości zrzutu danych z pominięciem wykonania kodu odpowiedzialnego za zabezpieczenie tych danych. Pod uwagę należy także wziąć konieczność fizycznego dostępu do karty przez potencjalnego atakującego. W praktyce bardzo utrudnia to możliwości kopiowania aktualnego stanu pamięci karty. Czynniki te minimalizują ryzyko przeprowadzenia udanego ataku i przejęcia chronionych danych.

Rozdział 5

Wnioski i zalecenia wdrożeniowe

Programowanie aplikacji na karty elektroniczne to bardzo ciekawe zagadnienie otwierające szerokie możliwości tworzenia systemów o wysokim standardzie bezpieczeństwa danych i wygody użytkowania. Styl pisania kodu może miejscami przypominać pewne cechy programowania w językach niższego poziomu na różne platformy wbudowane, jednakże użycie języka JavaCard daje możliwości wykorzystania wielu udogodnień oferowanych przez mechanizmy programowania obiektowego. Należy jednak pamiętać o istniejących ograniczeniach i zasadach pisania programów bezpiecznych w sensie *safe* oraz *secure*. Wykonany projekt jest realizacją istniejącego rozwiązania, które docelowo będzie integrowane na kartach będących jednocześnie blankietami Elektronicznej Legitymacji Studenckiej. Pozwoli to na korzystanie ze zniżek oraz dostęp do platformy OSOZ przy użyciu jednej karty.

Niewątpliwie największym wyzwaniem i jednocześnie aspektem wymagającym większego nakładu pracy są funkcje związane z biometrią. Aby przygotować algorytm skutecznie wyznaczający odpowiednie podobieństwo między dwoma odciskami palców potrzebne są solidne podstawy teoretyczne i znajomość kilku podejść praktycznych. Wiedzę taką można pozyskać studiując co najmniej kilka wartościowych artykułów naukowych z tej dziedziny. Dodatkowym utrudnieniem była niewielka moc obliczeniowa karty i ograniczone funkcje oferowane przez API JavaCard (głównie brak funkcji realizujących przydatne działania matematyczne). Algorytm *Match-on-card* w obecnej wersji działa zbyt wolno i znacząco przekracza założony próg 5 sekund czasu przetwarzania. Zaleceniem wdrożeniowym jest skorzystanie z istniejących rozwiązań, które komunikowałyby się z apletem poprzez interfejsy *Shareable* i udostępniałyby wynik uwierzytelnienia biometrycznego dla apletu jOsoz. Inną drogą rozwoju jest udoskonalenie zaproponowanego algorytmu o poprawki wydajnościowe oraz modyfikacje funkcji oceniających przy zachowaniu obecnej architektury projektu.

Wdrażając projekt należy upewnić się także, że wykorzystane mechanizmy kryptograficzne zostały odpowiednio sparametryzowane, aby funkcje udostępniane przez kartę mogły zintegrować się z innymi elementami systemu. Konieczna jest także weryfikacja zgodności apletu z urządzeniami typu PINPAD.

Ze względów formalnych projekt nie został wdrożony. Pomimo tego faktu, oraz niespełnienia przez algorytm biometryczny wymogu co do założonego czasu odpowiedzi cel pracy magisterskiej został ostatecznie osiągnięty. Opracowano autorskie rozwiązanie integrujące w sobie funkcje karty OSOZ oraz biometrykę wraz z aplikacją demonstrującą jego wykorzystanie. Dwie karty z wgranym apletem oraz pliki wykonywalne aplikacji de-

monstracyjnej zostały wysłane do siedziby firmy w Katowicach z ofertą dalszego rozwoju projektu.

Podsumowując, do zastosowań komercyjnych projekt wymaga poprawek w funkcjach związanych z biometrią i przeprowadzenia intensywnych testów w środowisku systemu OSOZ. Na zakończenie niniejszej pracy, autor wyraża nadzieję na praktyczne wykorzystanie wyników w niej uzyskanych.

Dodatek A

Przypadki użycia

Przypadki użycia ułożone zostały dla aplikacji demonstracyjnej przy założeniu obecności czytnika kart oraz karty inteligentnej z wgranym apletem.

PU-1 - Personalizacja karty

Główny scenariusz:

1. Użytkownik wybiera opcję personalizacji karty.
2. Użytkownik podaje hasło karty (PUK).
3. System personalizuje kartę.
4. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest spersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.

PU-2 - Sprawdzenie stanu personalizacji karty

Główny scenariusz:

1. Użytkownik wybiera opcję sprawdzenia stanu apletu.
2. System wyświetla informację o stanie apletu.
3. Koniec przypadku użycia.

PU-3 - Sprawdzenie stanu inicjalizacji PINu

Główny scenariusz:

1. Użytkownik wybiera opcję sprawdzenia stanu inicjalizacji PINu.
2. System wyświetla informację o stanie PINu.
3. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.

PU-3 - Sprawdzenie stanu inicjalizacji biometryki

Główny scenariusz:

1. Użytkownik wybiera opcję sprawdzenia stanu inicjalizacji biometryki.

2. System wyświetla informację o stanie odcisku palca.
3. Koniec przypadku użycia.
Rozszerzenia:
2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.

PU-3 - Sprawdzenie stanu inicjalizacji biometryki
Główny scenariusz:
1. Użytkownik wybiera opcję sprawdzenia stanu inicjalizacji biometryki.
2. System wyświetla informację o stanie biometryki.
3. Koniec przypadku użycia.
Rozszerzenia:
2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.

PU-4 - Sprawdzenie stanu uwierzytelnienia PINem w sesji
Główny scenariusz:
1. Użytkownik wybiera opcję sprawdzenia stanu uwierzytelnienia PINem w sesji.
2. System wyświetla informację o stanie uwierzytelnienia PINem w sesji.
3. Koniec przypadku użycia.
Rozszerzenia:
2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Karta posiada niezainicjowany PIN.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.

PU-5 - Sprawdzenie stanu uwierzytelnienia biometrycznego w sesji
Główny scenariusz:
1. Użytkownik wybiera opcję sprawdzenia stanu uwierzytelnienia biometrycznego w sesji.
2. System wyświetla informację o stanie uwierzytelnienie biometrycznego w sesji.
3. Koniec przypadku użycia.
Rozszerzenia:
2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Karta nie posiada wgranego odcisku palca.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.

PU-6 - Odczytanie numeru OSOZ z karty

Główny scenariusz:

1. Użytkownik wybiera opcję odczytania numeru OSOZ z karty inteligentnej

2. System wyświetla numer karty.

3. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.

2.A.1. System wyświetla odpowiedni komunikat.

2.A.2. Koniec przypadku użycia.

PU-7 - Wgranie odcisku palca na kartę

Główny scenariusz:

1. Użytkownik wybiera opcję wgrania odcisku na kartę.

2. System prosi użytkownika o kilkukrotne przyłożenie palca do czytnika.

3. System wgrywa odcisk palca na kartę.

4. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.

2.A.1. System wyświetla odpowiedni komunikat.

2.A.2. Koniec przypadku użycia.

2.B. Karta posiada wgrany odcisk palca.

2.B.1. System wyświetla odpowiedni komunikat.

2.B.2. Koniec przypadku użycia.

3.A. Odcisk palca posiada nieprawidłowy format lub został źle przesłany do karty.

3.A.1. System wyświetla odpowiedni komunikat.

3.A.2. Koniec przypadku użycia.

PU-8 - Weryfikacja biometryczna

Główny scenariusz:

1. Użytkownik wybiera opcję weryfikacji biometrycznej.

2. System prosi użytkownika o kilkukrotne przyłożenie palca do czytnika.

3. System przesyła odcisk palca na kartę.

4. System informuje o wyniku uwierzytelnienia.

5. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.

2.A.1. System wyświetla odpowiedni komunikat.

2.A.2. Koniec przypadku użycia.

2.B. Karta nie posiada wgranego odcisku palca.

2.B.1. System wyświetla odpowiedni komunikat.

2.B.2. Koniec przypadku użycia.

2.C. Odcisk palca jest zablokowany.

2.C.1. System wyświetla odpowiedni komunikat.

2.C.2. Koniec przypadku użycia.

2.D. Użytkownik uwierzytelnił się biometrycznie w sesji.

2.D.1. System wyświetla odpowiedni komunikat.

2.D.2. Koniec przypadku użycia.
3.A. Odcisk palca posiada nieprawidłowy format lub został źle przesłany do karty.
3.A.1. System wyświetla odpowiedni komunikat.
3.A.2. Koniec przypadku użycia.

PU-9 - Zainicjowanie PINu
Główny scenariusz:
1. Użytkownik wybiera opcję inicjalizacji PINu.
2. System prosi użytkownika o nowy PIN oraz jego potwierdzenie.
3. System ustawia PIN.
4. Koniec przypadku użycia.
Rozszerzenia:
2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Karta posiada zainicjowany PIN.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.
3.A. Wartości nowego PINu oraz jego potwierdzenia nie są identyczne.
3.A.1. System wyświetla odpowiedni komunikat.
3.A.2. Koniec przypadku użycia.

PU-10 - Weryfikacja PINem
Główny scenariusz:
1. Użytkownik wybiera opcję weryfikacji PINem.
2. System prosi użytkownika o podanie PINu.
3. System informuje o wyniku uwierzytelnienia.
4. Koniec przypadku użycia.
Rozszerzenia:
2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Karta nie posiada zainicjowanego PINu.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.
2.C. Użytkownik uwierzytelnił się PINem w sesji.
2.C.1. System wyświetla odpowiedni komunikat.
2.C.2. Koniec przypadku użycia.
2.D. PIN jest zablokowany.
2.D.1. System wyświetla odpowiedni komunikat.
2.D.2. Koniec przypadku użycia.

PU-11 - Zmiana PINu
Główny scenariusz:
1. Użytkownik wybiera opcję zmiany PINu.

2. System prosi użytkownika o podanie aktualnego PINu.
3. System prosi użytkownika o podanie nowego PINu oraz jego potwierdzenie.
4. System ustawia nowy PIN.
5. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Karta nie posiada zainicjowanego PINu.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.
2.C. PIN jest zablokowany.
2.C.1. System wyświetla odpowiedni komunikat.
2.C.2. Koniec przypadku użycia.
2.D. Użytkownik uwierzytelnił się PINem w sesji.
2.D.1. Przejście do kroku 3.
4.A. Wprowadzony aktualny PIN jest niepoprawny.
4.A.1. System wyświetla odpowiedni komunikat.
4.A.2. Koniec przypadku użycia.
4.A. Wprowadzony nowy PIN jest różny od PINu potwierdzającego.
4.A.1. System wyświetla odpowiedni komunikat.
4.A.2. Koniec przypadku użycia.

PU-12 - Odblokowanie odcisku palca

Główny scenariusz:

1. Użytkownik wybiera opcję odblokowania odcisku palca.
2. System prosi użytkownika o podanie kodu PUK.
3. System odblokowuje odcisk palca.
4. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Karta nie posiada zainicjowanego odcisku palca.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.
3.A. Użytkownik podał nieprawidłowy kod PUK.
3.A.1. System wyświetla odpowiedni komunikat.
3.A.2. Koniec przypadku użycia.
3.A. Odcisk palca nie jest zablokowany.
3.A.1. System wyświetla odpowiedni komunikat.
3.A.2. Koniec przypadku użycia.

PU-13 - Generowanie kodu autoryzacyjnego OSOZ

Główny scenariusz:

1. Użytkownik wybiera opcję wygenerowania kodu autoryzacyjnego.
2. System wyświetla kod wygenerowany przez kartę i informuje o jego weryfikacji.
3. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
2.B. Sygnatura kodu nie zgadza się z kluczem publicznym karty.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Koniec przypadku użycia.

PU-13 - Podpisywanie wiadomości kluczem karty

Główny scenariusz:

1. Użytkownik wybiera opcję elektronicznego podpisania wiadomości.
2. System prosi użytkownika o wiadomość do podpisania.
3. System wyświetla wygenerowany podpis i go weryfikuje.
4. Koniec przypadku użycia.

Rozszerzenia:

2.A. Karta jest niespersonalizowana.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Koniec przypadku użycia.
3.B. Sygnatura wiadomości nie zgadza się z kluczem publicznym karty.
3.B.1. System wyświetla odpowiedni komunikat.
3.B.2. Koniec przypadku użycia.

Dodatek B

Płyta CD

- Projekt aplikacji demonstracyjnej wykonany w środowisku Eclipse
- Projekt apletu ze skryptem kompilującym i niezbędnymi bibliotekami wykonany w środowisku Eclipse
- Skompilowany plik .cap, gotowy do wgrania na kartę

Bibliografia

[1] W. Rankl, W. Effing, *Smart Card Handbook*, John Wiley & Sons Ltd., 2003

[2] P. Nazimek, *Inżynieria programowania kart inteligentnych*, Politechnika Warszawska, Warszawa, 2005

[3] *Funkcjonalność karty ELS*, Międzyuczelniane Centrum Personalizacji Legitymacji studenckiej, <http://www.mcp.put.poznan.pl/?q=funkcje+ELS> (dostęp na dzień 25.05.2014)

[4] J. Tomczak, *Aplikacja biblioteczna jElib*, Politechnika Poznańska, Poznań, 2009

[5] Ł. Guzik, *Procesorowa Biometryczna Karta Pacjenta*, Almanach "Polskie Karty", 2010

[6] Biometric Consortium Interoperability, Assurance, and Performance Working Group, *Java Card Biometric API White Paper* (Working Document) Version 1.1, Document #: 02-0016, 2002

[7] *Java Card 2.2.1 API Specification*, <http://www.win.tue.nl/pinpasjc/docs/apis/jc221/> (dostęp dnia 31.05.2014)

[8] *ISO 7816 Part 4: Interindustry Commands for Interchange*, <http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4.aspx> (dostęp dnia 31.05.2014)

[9] *Java Card & STK Applet Development Guidelines 2.0*, Gemalto, 2009

[10] *Package javax.smartcardio API Specification*, <http://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html> (dostęp dnia 31.05.2014)

[11] *Java Platform, Standard Edition 7 API Specification*, <http://docs.oracle.com/javase/7/docs/api/> (dostęp dnia 31.05.2014)

[12] *Fingerprint SDK 2009 Supported Fingerprint Readers*, <http://www.griaulebiometrics.com/en-us/fingerprint_sdk/supported_readers> (dostęp dnia 31.05.2014)

[13] *Global Platform Card Specification Version 2.1.1*, GlobalPlatform Inc., 2003

[14] *Griaule products installers*, <http://www.griaulebiometrics.com/en-us/downloads> (dostęp dnia 31.05.2014)

[15] *Zvetco P5500 Fingerprint Device Specifications*, <http://www.zvetcobiometrics.com/Products/P5500/tech_specs.php> (dostęp dnia 31.05.2014)

[16] *Zvetco P4000 Fingerprint Device Specifications*, <http://www.zvetcobiometrics.com/Products/Legacy/P4000/overview.php> (dostęp dnia 31.05.2014)

[17] *Global Platform for Java SmartCardIO*, <http://sourceforge.net/projects/gpj/> (dostęp na 31.05.2014)

[18] *Implementation of GlobalPlatform smart card specification*, <http://sourceforge.net/p/globalplatform/wiki/GPShell/> (dostęp dnia 31.05.2014)

[19] *Java Card 2.2.1 API Specification*, <http://www.win.tue.nl/pinpasjc/docs/apis/jc222/> (dostęp dnia 2.06.2014)

[20] W. Mostowski, *Java Card Non-Atomic Methods*, Radboud Universiteit Nijmegen, 2006

[21] W. Mostowski, E. Hubbers, E. Poll, *Tearing Java Cards*, Radboud Universiteit Nijmegen, 2006

[22] A. Bazen and S. Gerez, *Fingerprint matching by thin-plate spline modelling of elastic deformations*, Pattern Recognition, vol. 36, pp.1859-1867, 2003.

[23] B. Dorizzi, R. Cappelli, M. Ferrara, D. Maio, D. Maltoni, N. Houmani, S. Garcia-Salicetti and A. Mayoue, *Fingerprint and On-Line Signature Verification Competitions at ICB 2009*, in proceedings International Conference on Biometrics (ICB), Alghero, Italy, pp.725-732, June 2009.

[24] *FVC-onGoing Published Results*, <https://biolab.csr.unibo.it/fvcongoing/UI/Form/PublishedAlgs.aspx> (dostęp na 9.06.2014)

[25] *Ustawa z dnia 29 sierpnia 1997 r. o ochronie danych osobowych*, <http://isip.sejm.gov.pl/DetailsServlet?id=WDU19971330883> (dostęp dnia 9.06.2014)

[26] *Rozporządzenie Ministra Nauki i Szkolnictwa Wyższego z dnia 2 listopada 2006 r. w sprawie dokumentacji przebiegu studiów*, <http://isap.sejm.gov.pl/DetailsServlet?id=WDU20062241634> (dostęp dnia 13.06.2014)

[27] *Biometrics History*, <http://www.biometrics.gov/documents/biohistory.pdf> (dostęp dnia 14.06.2014)

[28] *Jak OSOZ wspiera codzienną pracę lekarza?*, <http://www.osoz.pl/osoz/web/osoz-cms/wspieranie-pracy> (dostęp dnia 14.06.2014)

[29] M. Leszner, *Projekt i implementacja uniwersalnej aplikacji do obsługi kart elektronicznych*, Politechnika Poznańska, Poznań, 2012

[30] *Biometric data interchange formats; Part 2: Finger minutiae data* (committee draft), ISO/IEC, 2005