Politechnika Poznańska
Wydział Informatyki
Instytut Informatyki

Praca dyplomowa magisterska

KARTA ELEKTRONICZNA JAKO HARDWARE SECURITY MODULE (HSM)

Krystian Wiśniewski

Promotor
dr hab. inż. Marek Mika

Opiekun
mgr inż. Marek Gosławski

Poznań, 2019

Spis treści

Rozdział 1

Wstęp

Wiele firm wykorzystuje dane, które wymagają ochrony na bardzo wysokim poziomie. Są to informacje takie jak: krytyczne dane biznesowe, dane kart płatniczych, dane osobowe. Z tego powodu urządzenia zabezpieczające te informacje odgrywają istotną rolę w naszym codziennym życiu. W celu ich ochrony wykorzystuje się między innymi sprzętowe moduły bezpieczeństwa (*HSM*, ang. *Hardware Security Module*). Są to odporne na manipulacje urządzenia, które wzmacniają procedury związane z ochroną danych poprzez generowanie kluczy, szyfrowanie i deszyfrowanie danych oraz tworzenie i weryfikowanie podpisów cyfrowych.

Jednym z powszechnie używanych, zaufanych urządzeń, są karty elektroniczne. Są one stosowane między innymi w usługach bankowych. Podczas transakcji sprzedawca chce mieć pewność, że kupujący jest prawowitym właścicielem karty, a nie oszustem wykorzystującym skradzione numery kart bankowych. Przed dokonaniem płatności trzeba również potwierdzić, że sprzedawca jest naprawdę tym, za kogo się podaje. W związku z tym obie strony w tej transakcji muszą wzajemnie uwierzytelniać swoją tożsamość. Obie strony potrzebują poufności i integralności. Chcą one mieć pewność, że transakcje są chronione przed szpiegowaniem, ujawnianiem oraz przechwytywaniem informacji. Dlatego tak ważna jest ochrona klucza w urządzeniu. Obowiązuje ona przez cały okres użytkowania klucza, od jego wygenerowania, aż do usunięcia. Jeśli w pewnym momencie klucz nie jest chroniony, to nie można być pewnym, że nie wykonano kopii, która zagraża zachowaniu bezpieczeństwa.

1.1 Cel pracy

Celem pracy było sprawdzenie możliwości zastosowania karty elektronicznej jako sprzętowy moduł bezpieczeństwa. W ramach pracy zostały przeanalizowane dostępne rozwiązania na rynku, które mogą zostać wykorzystane przez Międzyuczelniane Centrum Personalizacji Legitymacji Studenckiej (*MCPLS*). Kolejnym etapem pracy był projekt oraz implementacja oprogramowania współpracującego z kartami elektronicznymi w oparciu o wcześniej zgromadzone informacje.

Rozdział 2

Standardy dla HSM

2.1 Karty elektroniczne

Karty elektroniczne [26] to urządzenia, które wykorzystywane są do przechowywania i przetwarzania danych. Posiadają wbudowany mikroprocesor lub układ pamięci, który w połączeniu z czytnikiem kart ma moc obliczeniową umożliwiającą obsługę wielu różnych aplikacji. Oprogramowanie znajdujące się w ich pamięci umożliwia bezpieczną identyfikację oraz uwierzytelnianie posiadacza, a także osób trzecich, które chcą uzyskać dostęp do karty. Do uwierzytelniania można użyć kodu PIN lub danych biometrycznych. Umożliwiają one również bezpieczne przechowywanie danych na karcie i zabezpieczają komunikację za pomocą szyfrowania. Karty stanowią przenośny, bezpieczny oraz łatwy w użyciu nośnik danych.

2.1.1 Charakterystyka

Karty elektroniczne mogą posiadać dwa różne rodzaje interfejsów: stykowy i bezstykowy. Stykowe karty są wkładane do czytnika kart, zapewniając fizyczny kontakt z czytnikiem. Natomiast karty bezstykowe mają wbudowaną wewnętrzną antenę, która umożliwia komunikację z czytnikiem bez fizycznego kontaktu. Na rynku istnieją również karty hybrydowe, które posiadają wbudowane obydwa typy interfejsów.

Charakterystyki stykowych kart elektronicznych są określone przez międzynarodową normę ISO/IEC 7816 [28] zarządzaną wspólnie przez Międzynarodową Organizację Normalizacyjną (ISO) i Międzynarodową Komisję Elektrotechniczną (IEC). Natomiast standardem dla kart bezstykowych jest norma ISO/IEC 14443[28]. Większość kart bezstykowych pracuje w paśmie 13,56 MHz oraz umożliwia komunikację w bliskiej odległości do 10 cm.

Rozmiary kart elektronicznych określone są przez międzynarodową normę ISO/IEC 7810 [28]. Dla kart elektronicznych w formacie ID–1 [26], rozmiar wynosi 85,60 × 53,98 mm, a karta posiada zaokrąglone narożniki o promieniu 2,88-3,48 mm. Format ten jest stosowany w przypadku kart PET, PVC, poliwęglanowych, a nawet pełnych kart metalowych.

2.1.2 Bezpieczeństwo

Karty elektroniczne [37] wykorzystywane jako aktywne urządzenia obliczeniowe pozwalają zapewnić ogromne korzyści płynące z przenośnego oraz bezpiecznego przechowywania danych i wartości. Z tego powodu zostały zaprojektowane tak, aby były odporne na manipulacje. Dane zapisane na karcie można wykorzystywać do zidentyfikowania oraz uwierzytelniania użytkownika w celu przeprowadzenia odpowiednich operacji. Dostęp do określonych informacji zawartych na karcie jest kontrolowany przez system operacyjny karty oraz ustawione wstępnie uprawnienia przez wystawcę karty.

Aby wspierać procesy odpowiadające za bezpieczeństwo, takie jak na przykład uwierzytelnianie, karty elektroniczne są wyposażone w koprocesor kryptograficzny. Może on skutecznie wykonywać operacje kryptograficzne dla symetrycznych, jak i asymetrycznych algorytmów kryptograficznych.

Dostęp do danych zawartych w pamięci karty wymaga odpowiedniego rodzaju kontroli dostępu [10]. Osiąga się to zazwyczaj poprzez wykorzystanie kodu PIN (Personal Identification Number). Towarzyszy mu zazwyczaj kod PUK (Personal Unlocking Key) zwany również SO PIN(Security Officer). Za bezpieczeństwo tego drugiego odpowiada pracownik ds. bezpieczeństwa (ang. *Security Officer*). Jest to osoba lub rola, która jest odpowiedzialna za proces zarządzania kartami elektronicznymi.

2.2 Sprzętowy moduł bezpieczeństwa

Sprzętowy moduł bezpieczeństwa [19] to urządzenie, które zostało zaprojektowane do ochrony kluczy kryptograficznych przez cały ich cykl życia. Odpowiada on za bezpieczne zarządzanie, przetwarzanie i przechowywanie kluczy kryptograficznych wewnątrz utwardzonego (ang. *hardening*), odpornego na manipulacje urządzenia. Umożliwia przeprowadzanie operacji kryptograficznych takich jak: szyfrowanie i deszyfrowanie danych oraz tworzenie i weryfikację podpisów cyfrowych.

2.2.1 Zastosowanie

Moduły HSM [14] wykorzystuje się w celu zapewnienia silnego uwierzytelniania oraz wykonywania operacji kryptograficznych [19]. Sprzętowe moduły bezpieczeństwa są coraz częściej kupowane oraz wdrażane przez przedsiębiorstwa w celu wzmocnienia infrastruktury kryptograficznej. Doskonale sprawdzają się w zabezpieczaniu kluczy kryptograficznych oraz świadczeniu usług szyfrowania, deszyfrowania, uwierzytelniania i podpisywania cyfrowego dla szerokiego zakresu zastosowań. Wykorzystywane są w różnych aplikacjach biznesowych, w tym infrastrukturze klucza publicznego (*PKI*, ang. *Public Key Infrastructure*) do ochrony klucza szyfrowania SSL/TLS, podpisywania kodów, podpisu cyfrowego i łańcucha blokowego (ang. *Blockchain*).

2.2.2 Bezpieczeństwo

Ze względu na krytyczną rolę, jaką moduły HSM odgrywają w zabezpieczaniu aplikacji i infrastruktury, są one zazwyczaj certyfikowane zgodnie z międzynarodowymi standardami, takimi jak Common Criteria lub FIPS 140 [18]. Najwyższym osiągalnym poziomem certyfikacji bezpieczeństwa FIPS 140 jest poziom bezpieczeństwa 4, natomiast w przypadku CC jest to EAL7 [34]. W przypadku zastosowania w operacjach finansowych, bezpieczeństwo modułu HSM jest często sprawdzane pod kątem wymogów określonych przez Radę ds. standardów bezpieczeństwa branży kart płatniczych (ang. *Payment Card Industry Security Standards Council*) [32]. Zastosowanie odpowiednich standardów daje użytkownikom pewność, że projekt i implementacja produktu oraz algorytmy kryptograficzne są prawidłowe. Bezpieczeństwo zależy od właściwych narzędzi, procesów oraz polityk wykorzystywania tych urządzeń. Sprzętowy moduł bezpieczeństwa jest bezpieczniejszy niż oprogramowanie, ponieważ jest mniej podatny na awarie i uszkodzenia systemu, spowodowane na przykład przez wirusy. Zapewnia wyższy poziom bezpieczeństwa niż niezabezpieczone nośniki, takie jak zapasowe dyskietki, płyty, karty pamięci, ponieważ te ostatnie mogą być łatwo usunięte lub skopiowane. HSM pozwala na kontrolowany dostęp do przechowywanych kluczy. Kody dostępu mogą być dystrybuowane pomiędzy kilka osób, które muszą współpracować, aby uzyskać dostęp do klucza głównego.

2.3 Standardy bezpieczeństwa

Sprzętowe moduły bezpieczeństwa zgodne z obowiązującymi normami oraz standardami zapewniają wiarygodną ochronę przed naruszeniem danych oraz aplikacji. Moduły HSM są dzięki temu odporne na manipulacje i oferują najwyższy poziom bezpieczeństwa.

2.3.1 Common Criteria

Common Criteria (*CC*) [24] jest uznaną, międzynarodową normą ISO (ISO/IEC 15408) stosowaną przez rządy i inne organizacje do oceny bezpieczeństwa i zapewnienia bezpieczeństwa produktów technologicznych. Common Criteria daje pewność, że proces specyfikacji, wdrażania i oceny produktu bezpieczeństwa komputerowego został przeprowadzony w rygorystyczny i ustandaryzowany sposób. Common Criteria składa się z 3 części:

- „Introduction and General Model" — jest wprowadzeniem do CC. Określa ogólne pojęcia i zasady oceny bezpieczeństwa IT oraz przedstawia ogólny model oceny,
- „Security Functional Components" — określa wymagania w zakresie bezpieczeństwa,
- „Security Assurance Components" — określa kryteria oceny wymogów bezpieczeństwa oraz przedstawia siedem wstępnie zdefiniowanych zestawów oceny wiarygodności, które nazywane są ang. *Evaluation Assurance Levels* (*EAL*).

Kluczowymi pojęciami *CC* są:

- *TOE* (ang. *Target of Evaluation*) — oprogramowanie (ang. *software*), oprogramowanie układowe (ang. *firmware*) lub sprzęt (ang. *hardware*) będący przedmiotem oceny,
- *PP* (ang. *Protection Profile*) — dokument, zazwyczaj tworzony przez użytkownika lub społeczność użytkowników. Określa wymogi bezpieczeństwa dla danej klasy urządzeń zabezpieczających istotnych dla danego użytkownika w konkretnym celu.
- *ST* (ang. *Security Target*) — zestaw wymogów bezpieczeństwa, który zawiera obszerne, specyficzne dla danego produktu informacje. Stanowi podstawę do porozumienia pomiędzy deweloperami, konsumentami i oceniającymi bezpieczeństwo.
- *SARs* (ang. *Security Assurance Requirements*) — opisy działań podjętych podczas opracowywania i oceny produktu w celu zapewnienia zgodności z deklarowaną funkcjonalnością bezpieczeństwa.
- *EAL* (ang. *Evaluation Assurance Level*) — numeryczna klasyfikacja opisująca dogłębność oceny. Każdy *EAL* odpowiada pakietowi wymogów bezpieczeństwa (*SAR*), który obejmuje pełny rozwój produktu o określonym poziomie rygorystyczności. *CC* wymieniają siedem poziomów, przy czym *EAL* 1 jest najbardziej podstawowym, a *EAL* 7 jest najbardziej rygorystyczny. Wyższe *EAL* nie oznacza wyższego bezpieczeństwa, a jedynie potwierdzenie, że *TOE* został szerzej zweryfikowany.

2.3.2 FIPS 140-2

Federalny Standard Przetwarzania Informacji (*FIPS*, ang. *The Federal Information Processing Standard*) Publikacja 140-2, jest amerykańskim rządowym standardem bezpieczeństwa komputerowego stosowanym do zatwierdzania modułów kryptograficznych. Tytuł to „Security Requirements for Cryptographic Modules". Publikacja ta ukazała się 25 maja 2001 r. i została ostatnio zaktualizowana 3 grudnia 2002 r. Standard ten określa wymogi bezpieczeństwa, które powinien spełnić moduł kryptograficzny wykorzystywany w ramach systemu bezpieczeństwa chroniącego dane istotne, ale nie zastrzeżone (*SBU*, ang. *Sensitive but unclassified*), zwane dalej "informacjami szczególnie chronionymi (ang. *sensitive information*)".

Poziomy ochrony

Standard ten zapewnia cztery wzrastające poziomy ochrony: Poziom 1, Poziom 2, Poziom 3 i Poziom 4. Poziomy te obejmują szeroki zakres możliwych aplikacji i środowisk, w których można stosować moduły kryptograficzne. Wymagania bezpieczeństwa obejmują obszary związane z bezpiecznym projektowaniem oraz wdrażaniem modułu kryptograficznego. Obszary te uwzględniają:

- specyfikację modułów kryptograficznych, porty i interfejsy modułów kryptograficznych,
- role, usługi i uwierzytelnianie,
- bezpieczeństwo fizyczne,
- środowisko operacyjne,
- zarządzanie kluczami kryptograficznymi,
- zaburzenia oraz kompatybilność elektromagnetyczną (*EMI/EMC*),
- testy własne,
- łagodzenie skutków innych ataków.

Skrócony opis każdego poziomu bezpieczeństwa [42] jest następujący:

- Poziom 1: Jest to najbardziej podstawowy poziom bezpieczeństwa, który wymaga wykorzystania tylko jednego zatwierdzonego algorytmu lub funkcji bezpieczeństwa, ale nie wymaga fizycznej ochrony HSM.
- Poziom 2: Wymaga wykorzystania w HSM mechanizmu zabezpieczającego dowody manipulacji i uwierzytelniania opartego na rolach.
- Poziom 3: Wymaga zabezpieczenia przed sabotażem wraz z dowodem sabotażu i uwierzytelniania opartego na tożsamości.
- Poziom 4: Najbardziej bezpieczny poziom obejmuje rozpoznawanie i łagodzenie napaści związanych z bezpieczeństwem fizycznym oraz warunkami środowiskowymi, zapewniając kompleksowe bezpieczeństwo HSM nawet w fizycznie niezabezpieczonym środowisku.

2.4 Interfejsy programowania aplikacji

Interfejs programowania aplikacji (*API*, ang. *Application Program Interface*) to zestaw procedur, protokołów i narzędzi do tworzenia aplikacji. Określa, w jaki sposób komponenty oprogramowania powinny ze sobą współdziałać. Dodatkowo API są używane podczas programowania komponentów graficznych interfejsu użytkownika (GUI). Dobre API ułatwia opracowanie programu poprzez dostarczenie wszystkich elementów konstrukcyjnych.

2.4.1 PKCS#11

PKCS#11 [30] jest jednym ze Standardów Kryptografii Klucza Publicznego (ang. *Public-Key Cryptography Standards*). Standard ten opracowany został po raz pierwszy przez RSA Laboratories we współpracy z przedstawicielami przemysłu, nauki i rządów. Obecnie jest otwartym standardem zarządzanym przez Komitet Techniczny OASIS PKCS#11 (ang. *OASIS PKCS#11 Technical Committee*). Standard PKCS#11 określa interfejs programowania aplikacji (*API*), zwany „Cryptoki", dla urządzeń przechowujących informacje kryptograficzne i wykonujących funkcje kryptograficzne. Stosuje on podejście obiektowe oraz jest niezależny technologicznie, dzięki czemu umożliwia współpracę różnych urządzeń. Pozwala na współdzielenie zasobów pomiędzy różnymi aplikacjami i urządzeniami, prezentując im wspólny, logiczny interfejs urządzenia zwany „tokenem kryptograficznym". PKCS#11 przypisuje każdemu tokenowi identyfikator slotu. Aplikacja identyfikuje token, do którego chce uzyskać dostęp, podając odpowiedni identyfikator slotu.

2.4.2 Microsoft CryptoAPI

CryptoAPI [4] to kryptograficzny interfejs programowania aplikacji dołączany do systemów operacyjnych Microsoft Windows. Umożliwia twórcom aplikacji dodawanie uwierzytelniania, kodowania i szyfrowania do aplikacji opartych na systemie Windows. Nie wspiera on kryptografii krzywych eliptycznych.

2.4.3 Microsoft Cryptographic Service Provider

W systemach Microsoft Windows, dostawca usług kryptograficznych (*CSP*, ang. *Cryptographic Service Provider*) [5] jest biblioteką oprogramowania, która implementuje Microsoft CryptoAPI. Jest to niezależny moduł oprogramowania wykorzystywany przez aplikacje komputerowe, który realizuje algorytmy kryptograficzne do uwierzytelniania, kodowania i szyfrowania. Wybór dostawcy usług kryptograficznych decyduje o rodzaju, rozmiarze i sposobie przechowywania klucza. CSP zawiera implementacje standardów i algorytmów kryptograficznych.

2.4.4 Cryptography API: Next Generation

Cryptography API: Next Generation (*CNG*) [2] jest drugą generacją CryptoAPI. CNG [3] umożliwia zastąpienie istniejących dostawców algorytmów własnymi dostawcami oraz dodanie nowych algorytmów, gdy staną się one dostępne. Nowe API pozwola na stosowanie nowszych, silniejszych algorytmów szyfrowania i podpisywania. W wersji tej wprowadzono obsługę kryptografii krzywych eliptycznych. Został zaprojektowany jako zgodny z FIPS 140-2 Level 2. CNG spełnia wymagania Common Criteria poprzez przechowywanie i wykorzystanie kluczy w bezpiecznym środowisku. API integruje się z podsystemem kart elektronicznych dzięki wykorzystanie modułu Base Smart Card Cryptographic Service Provider (Base CSP). CNG jest obsługiwany od systemów Windows Server 2008 i Windows Vista.

2.4.5 Base Smart Card Cryptographic Service Provider

Microsoft Base Smart Card CSP [20] umożliwia sprzedawcom kart inteligentnych łatwiejsze wykorzystywanie kart w systemach Windows za pomocą lekkiego, zastrzeżonego modułu kart zamiast pełnego CSP. Implementuje wspólne funkcje, takie jak haszowanie, symetryczne oraz asymetryczne operacje, obsługa PIN. Sprzedawca musi zapewnić sterownik (zwany również modułem), który zapewnia wspólny interfejs współpracujący z Base Smart Card CSP.

2.5 OpenSC

OpenSC [11] zapewnia zestaw bibliotek i narzędzi do pracy z kartami elektronicznymi. Główny nacisk kładzie się na karty obsługujące operacje kryptograficzne. Zestaw ten ułatwia wykorzystanie ich w aplikacjach w celu zwiększenia bezpieczeństwa. Umożliwa operacje takie jak: uwierzytelnianie, szyfrowanie poczty oraz podpisy cyfrowe. OpenSC implementuje standardowe API do kart, np. PKCS#11 API, Windows' Smart Card Minidriver oraz macOS Tokend.

Rozdział 3

Dywersyfikacja kluczy

Dywersyfikacja klucza polega na pozyskaniu jednego lub większej liczby kluczy z wartości tajnej. Jako wartość początkowa może zostać użyty klucz główny (ang. *Master key*), hasło wprowadzone przez użytkownika lub hasło wygenerowane przy pomocy funkcji pseudolosowych. Jeśli jeden klucz zostanie złamany to problem dotyczy tylko danej karty, a nie całego systemu. Z punktu widzenia bezpieczeństwa, ważne jest, aby tajna wartość zawierała wystarczająco dużo elementów losowości, aby ograniczyć możliwości przeprowadzenia ataków wykorzystujących informacji o niej.

3.1 Bezpieczna komunikacja

Dokumentacja GlobalPlatform [13] definiuje pojęcie bezpiecznej komunikacji (ang. *Secure Communication*) dokładniej niż w normach zdefiniowanych przez standard ISO. Zwraca szczególną uwagę na bezpieczne przesyłanie poleceń oraz odpowiedzi APDU (ang. *Application Protocol Data Unit*), dzięki opcjonalnemu procesowi uwierzytelniania oraz możliwości tworzenia sesji Bezpiecznego Kanału (ang. *Secure Channel Session*).

Bezpieczny Kanał zapewnia trzy poziomy bezpieczeństwa:

- Uwierzytelnianie podmiotu (ang. *Entity authentication*) - jednostka wydająca kartę lub niebędąca kartą udowadnia innej jednostce swoją prawdziwość dzięki wykorzystaniu operacji kryptograficznych,
- Integralność i uwierzytelnianie (ang. *Integrity and authentication*) - jednostka otrzymująca dane potwierdza, że pochodziły one od jednostki uwierzytelnionej, ich kolejność jest prawidłowa i nie zostały zmienione,
- Poufność (ang. *Confidentiality*) - dane przekazywane przez jednostkę wysyłającą do jednostki otrzymującej nie są widoczne dla podmiotu nieuwierzytelnionego.

3.2 Bezpieczny Kanał

Bezpieczny Kanał (ang. *Secure Channel*) jest wykorzystywany do zapewnienia bezpiecznego kanału komunikacyjnego pomiędzy kartą a jednostką niebędącą kartą (hostem).

Sesja Bezpiecznego Kanału (ang. *Secure Channel Session*) jest podzielona na trzy kolejne fazy:

- Inicjacja (ang. *Secure Channel Initiation*) – wymiana niezbędnych informacji pomiędzy aplikacją na karcie a hostem, umożliwiających im wykonanie wymaganych funkcji kryptograficznych. Inicjacja zawsze obejmuje uwierzytelnianie hosta przez aplikację na karcie,
- Działanie (ang. *Secure Channel Operation*) – wymiana informacji pomiędzy aplikacją na karcie a hostem w ramach kryptograficznej ochrony sesji. Poziom zabezpieczeń może się różnić w zależności od Protokołu Bezpiecznego Kanału (ang. *Secure Channel Protocol*).
- Zakończenie (ang. *Secure Channel Termination*) – bezpieczne zakończenie następuje, gdy aplikacja na karcie lub host uzna, że dalsza komunikacja za pośrednictwem ustanowionego kanału nie jest wymagana lub dozwolona.

Jawna inicjacja Bezpiecznego Kanału

Jawne zainicjowanie sesji Bezpiecznego Kanału może zostać wykonane przez jednostkę niebędącą kartą (host) przy użyciu odpowiednich poleceń APDU lub przez aplikację na karcie przy użyciu odpowiedniego API.

Niejawna inicjacja Bezpiecznego Kanału

Sesja jest inicjowana przez Protkół Bezpiecznego Kanału (ang. *Secure Channel Protocol*) na karcie, bezpośrednio lub poprzez API, gdy pierwsza polecenie APDU zawiera ochronę kryptograficzną. Protokół niejawnie zna wymagany poziom bezpieczeństwa oraz może wiedzieć, które klucze mają być używane lub które mogły zostać wcześniej ustalone przez inną jednostkę poprzez odpowiednie polecenie APDU przed rozpoczęciem sesji bezpiecznego kanału.

Zamknięcie Bezpiecznego Kanału

Zamknięcie sesji może zostać wywołane przez aplikację na karcie używając API lub hosta poprzez użycie polecenia APDU.

Zamknięcie kanału powoduje, że wszystkie dane sesji są resetowane, a wszystkie wektory inicjujące (ICV, ang. *Initial Chaining Vector*) i klucze sesji są kasowane. Aktualny poziom zabezpieczeń zostaje ustawiony na wartość NO_SECURITY, a poziom zabezpieczeń sesji jest resetowany.

Zamknięcie sesji bezpiecznego kanału następuje po spełnieniu jednego z poniższych warunków:

- Sesja aplikacji na karcie zostaje zakończona, na przykład, gdy wybrana zostanie inna aplikacja na tym samym kanale logicznym dla danego interfejsu I/O karty.
- Powiązany kanał logiczny zostanie jawnie zamknięty

- Karta zostanie zresetowana (karty stykowe) lub zostanie dezaktywowana (karty bezstykowe) lub wyłączona, np. sesja karty zostanie zakończona.

Jeśli sesja Bezpiecznego Kanału zostanie przerwana, bieżący poziom zabezpieczeń jest ustawiony na wartość NO_SECURITY, poziom zabezpieczeń sesji nie jest resetowany, a stan błędu utrzymuje się aż do zamknięcia bezpiecznej sesji kanału.

Sesja Bezpiecznego Kanału zostaje przerwana, ale nie jest zamknięta w następujących przypadkach:

- Aplikacja na karcie otrzymuje pierwsze polecenie APDU z błędną ochroną kryptograficzną,
- Aplikacja na karcie otrzymuje polecenie APDU bez uzyskania wymaganej ochrony kryptograficznej ustawionej podczas inicjowania sesji.

3.2.1 Obsługa Protokołu Bezpiecznego Kanału

Istnieją dwa sposoby korzystania przez aplikację na karcie z Protokołu Bezpiecznego Kanału:

- Bezpośrednia - aplikacja posiada zestaw kluczy kanału bezpiecznego i w pełni implementuje protokół, np. Domeny Bezpieczeństwa (ang. *Security Domains*),
- Pośrednia - Aplikacja wykorzystuje usługi Domeny Bezpieczeństwa do obsługi protokołu. Dzięki temu aplikacja korzystająca z tych usług może być szyfrowana niezależnie od protokołu obsługiwanego przez kartę.

3.2.2 Uwierzytelnianie podmiotu

Uwierzytelnianie podmiotu niebędącego kartą osiągane jest poprzez proces inicjowania sesji Bezpiecznego Kanału, dzięki czemu karta otrzymuje potwierdzenie, że komunikuje się z jednostką uwierzytelnioną. Jeżeli którykolwiek z etapów procesu uwierzytelniania zakończy się niepowodzeniem, proces zostanie wznowiony. uwierzytelnianie jest zapewnione tylko na ograniczony czas i jest ważne tylko dla komunikatów w ramach tego bezpiecznego kanału. Sesja dotyczy ustanowienia i zakończenia Bezpiecznego Kanału. Jeśli sesja zostanie zamknięta z jakiegokolwiek powodu, host nie będzie dłużej uznawany za wiarygodny.

uwierzytelnianie można uzyskać przy wykorzystaniu kryptografii:

- Symetrycznej – z wykorzystaniem klucza symetrycznego Protokołu Bezpiecznego Kanału, np. SCP01, SCP02. Uwierzytelniony host jest jednostką, która zna tajne klucze niezbędne do zainicjowania sesji.
- Asymetrycznej - z wykorzystaniem klucza asymetrycznego Protokołu Bezpiecznego Kanału, np. SCP10. Uwierzytelniona jednostka, która posiada parę kluczy asymetrycznych i uzyskała certyfikat wydany przez organ uznany przez Domenę Bezpieczeństwa, może być skutecznie uwierzytelniona przez tę domenę.

3.2.3 Identyfikator Protokołu Bezpiecznego Kanału

Identyfikator Protokołu Bezpiecznego Kanału (ang. *Secure Channel Protocol Identifier*) określa, który protokół i zestaw usług bezpieczeństwa są wdrażane w Domenie Bezpieczeństwa. Wartości, które mogą zostać przypisane do identyfikatora protokołu:

- 00 – niedostępne,
- 01 do 7F - zarezerwowane do użytku przez GlobalPlatform:
  - 01 – Secure Channel Protocol 01,
  - 02 – Secure Channel Protocol 02,
  - 03 – Secure Channel Protocol 03,
  - 10 - Secure Channel Protocol 10,
- 80 do EF - zarezerwowane do użytku w indywidualnych schematach zarejestrowanych przez GlobalPlatfom,
- F0 do FF - zastrzeżone do użytku własnego, niezarejestrowane przez GlobalPlatform.

3.3 Secure Channel Protocol 02

Protokół Bezpiecznego Kanału 02 (ang. *Secure Channel Protocol 02*, SCP02) wykorzystuje algorytm szyfrowania Triple DES w trybie CBC (ang. *Cipher-Block Chaining*) z wektorem inicjalizacji (IV) składającym się z binarnych zer w celu zapewniane trzech poziomów bezpieczeństwa przez Bezpieczny Kanał.

3.3.1 Uwierzytelnianie podmiotu

Jednostki uwierzytelniają się wzajemnie, potwierdzając, że host posiada wiedzę na temat tych samych prywatnych informacji, co karta. Uwierzytelnianie jednostki jest osiągane poprzez proces inicjowania Bezpiecznego Kanału. W przypadku niepowodzenia któregokolwiek z etapów procesu uwierzytelniania, proces jest uruchamiany ponownie - generowane są nowe klucze sesji otrzymywane w procesie dywersyfikacji klucza matki.

3.3.2 Jawna inicjacja Bezpiecznego Kanału

Bezpieczny Kanał może zostać jawnie zainicjowany przez podmiot zewnętrzny używając poleceń INITIALIZE UPDATE oraz EXTERNAL AUTHENTICATE. Aplikacja może przekazać APDU do Domeny Bezpieczeństwa dzięki użyciu odpowiedniego API. Jednostka zewnętrzna pozwala określić, jaki poziom bezpieczeństwa jest wymagany dla danego Kanału Bezpieczeństwa oraz zastosować ten poziom dla wszystkich kolejnych komunikatów wymienianych aż do końca sesji. Może również wybrać numer wersji klucza (ang. *Key Version Number*), który będzie używany.

Bezpieczny Kanał jest zawsze inicjowany przez jednostkę zewnętrzną poprzez przekazanie wyzwania hosta (ang. *Host Challenge*) do karty – losowych, unikalnych danych dla danej sesji. Karta po otrzymaniu tego wyzwania, generuje własne wyzwanie karty (ang. *Card Challenge*) – również losowe dane unikalne dla danej sesji.

Karta, wykorzystując wewnętrzny licznik sekwencji i klucze statyczne, tworzy nowe tajne klucze sesji i generuje pierwszą wartość kryptograficzną, którą jest kryptogram karty (ang. *Card Cryptogram*) przy użyciu jednego z nowo utworzonych kluczy sesji.

Ten kryptogram karty wraz z licznikiem sekwencji, wyzwaniem karty, identyfikatorem Protokołu Bezpiecznego Kanału i innymi danymi jest przesyłany z powrotem do jednostki.

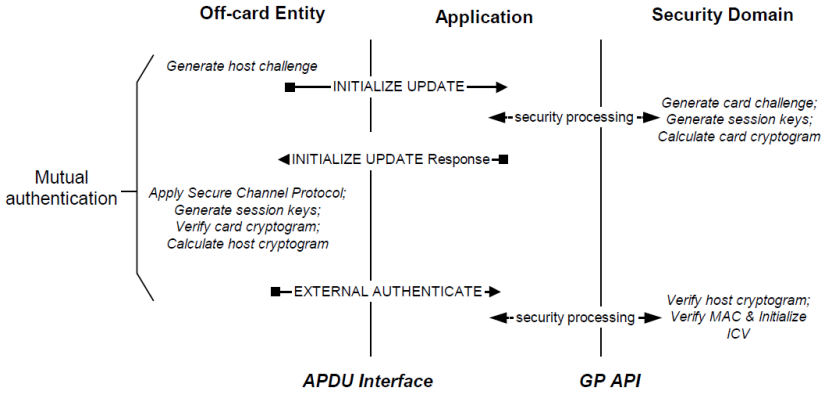
Podmiot dysponujący takimi samymi informacjami jak karta użyta do wygenerowania kryptogramu karty jest w stanie wygenerować te same klucze sesyjne i ten sam kryptogram karty. Następnie poprzez porównanie ich jest w stanie dokonać uwierzytelniania karty.

Jednostka wykorzystuje teraz podobny proces do utworzenia kryptogramu hosta (ang. *Host Cryptogram*), który ma być przekazany z powrotem na kartę.

Karta posiadając wszystkie te same informacje, które zostały wykorzystane przez hosta do wygenerowania kryptogramu hosta, powinna być w stanie wygenerować ten sam kryptogram, a następnie poprzez porównanie tych wartości, uwierzytelnić podmiot.

Host tworzy również MAC (ang. *Message Authentication Code*), który jest przekazywany z powrotem na kartę i weryfikowany przez nią. Jest to symetryczna kryptograficzna transformacja danych, która zapewnia uwierzytelnianie oraz integralność danych. Weryfikacja ta jest wykorzystywana przez kartę do stworzenia wstępnego wektora wiązania (ang. *Chaining Vector*) do weryfikacji C-MAC (MAC dołączony do polecenia APDU) i/lub R-MAC (MAC dołączony do odpowiedzi APDU). Po pomyślnym uwierzytelnianiu jednostki, karta zwiększa swój wewnętrzny Licznik Sekwencji Bezpiecznych Kanałów (ang. *Secure Channel Sequence Counter*).

Przykład inicjacji Bezpiecznego Kanału pomiędzy kartą a jednostką został przedstawiony na rysunku 3.1:



Rysunek 3.1: Inicjacja Bezpiecznego Kanału [13]

3.3.3 Niejawna inicjacja Bezpiecznego Kanału

Bezpieczny Kanał jest domyślnie inicjowany w momencie otrzymania pierwszego polecenia APDU, które zawiera ochronę kryptograficzną (C-MAC). Wymagany poziom bezpieczeństwa jest niejawnie znany zarówno przez kartę jak i host, jako integralna część polecenia. Host niejawnie wie, które klucze będą używane oraz zna aktualną wartość licznika sekwencji lub mógł pobrać te informacje przy użyciu polecenia GET DATA.

Host wykorzystując informacje na temat statycznych kluczy karty i jej Licznika Sekwencji Bezpiecznego Kanału, tworzy nowe tajne klucze sesji i generuje C-MAC. Bezpieczny Kanał jest inicjowany przez host poprzez dołączenie C-MAC do polecenia APDU.

Karta, po otrzymaniu pierwszego C-MAC, tworzy klucz sesji C-MAC przy użyciu swojego wewnętrznego statycznego klucza(y) karty oraz licznika sekwencji. Karta posiadając te same informacje, które zostały wykorzystane przez hosta do wygenerowania C-MAC, generuje ten sam MAC i wykonując porównanie umożliwia uwierzytelnianie hosta. Po pomyślnym uwierzytelnianiu jednostki, karta zwiększa swój wewnętrzny licznik sekwencji.

Do odszyfrowania poufnych danych, karta tworzy klucz sesji do szyfrowania danych przy użyciu statycznego klucza(y) karty oraz zwiększonej wartości licznika sekwencji. W celu wygenerowania R-MAC, karta tworzy klucz sesji R-MAC również przy użyciu statycznego klucza(y) oraz zwiększonej wartości licznika sekwencji.

3.3.4 Integralność poleceń

Podmiot otrzymujący zapewnia, że otrzymywane dane rzeczywiście pochodziły od uwierzytelnionego podmiotu wysyłającego w prawidłowej kolejności i nie zostały zmienione. Kod uwierzytelniania wiadomości (MAC) jest generowany przez zastosowanie wielu łańcuchowych operacji DES (przy użyciu klucza sesji wygenerowanego przed lub podczas otwierania Bezpiecznego Kanału) w postaci komunikatu APDU. MAC może zostać wygenerowany w następujący sposób:

- C-MAC dla komunikatów polecenia APDU (generowanych przez host);
- R-MAC dla komunikatów odpowiedzi APDU (generowanych przez kartę).

Jednostka odbierająca po otrzymaniu wiadomości zawierającej MAC, posługując się tym samym kluczem sesji, wykonuje tę samą operację. Następnie poprzez porównanie wygenerowanego przez nią MAC z MAC otrzymanym od jednostki wysyłającej jest zapewniona o integralności całego polecenia oraz odpowiedzi.

Integralność sekwencji poleceń APDU lub odpowiedzi przesyłanych do jednostki odbiorczej jest osiągana poprzez wykorzystanie MAC z bieżącego polecenia lub odpowiedzi jako początkowy wektor łańcuchowy (ICV) dla następnego polecenia lub odpowiedzi.

3.3.5 Poufność

Pole danych wiadomości jest szyfrowane przez zastosowanie wielu łańcuchowych operacji DES. Podczas tych operacji wykorzystywane są klucze sesji wygenerowanego podczas procesu inicjowania Bezpiecznego Kanału. Szyfrowaniu podlega całe pole danych, poleceń lub odpowiedzi.

3.3.6 Klucze kryptograficzne

Domena Bezpieczeństwa, w tym Domena Bezpieczeństwa Dostawcy powinna posiadać co najmniej jeden zestaw kluczy wykorzystywany przy inicjowaniu oraz używaniu Bezpiecznego Kanału. Zestaw kluczy powinien zawierać co najmniej jeden klucz DES o podwójnej długości (16 bajtów), czyli główny klucz Bezpiecznego Kanału, który jest używany tylko do generowania kluczy sesji podczas inicjowania Bezpiecznego Kanału.

Zestaw kluczy Domeny Bezpieczeństwa, w tym Domeny Bezpieczeństwa Dostawcy, może zawierać 3 klucze DES o podwójnej długości (16 bajtów):

- S-ENC - klucz szyfrowania, wykorzystywany do uwierzytelniania oraz enkrypcji
- S-MAC - klucz MAC do weryfikacji MAC oraz generowania DES
- DEK - klucz szyfrowania danych, do deszyfrowania poufnych danych, np. prywatnych informacji lub kluczy prywatnych. Klucze te są używane tylko do generowania kluczy sesyjnych podczas inicjowania Bezpiecznego Kanału.

3.3.7 Wektor inicjujący

Wektor inicjujący (ICV) używany podczas szyfrowania w trybie CBC używany jest do zapewnienia integralności wiadomości. Ma zawsze postać binarnych zer o całkowitej długości 8 bajtów. Inicjowany jest podczas tworzenia Bezpiecznego Kanału.

ICV dla Integralności Wiadomości

W przypadku wykorzystania jawnej inicjacji Bezpiecznego Kanału, SCP02 wymaga użycia MAC dla polecenia EXTERNAL AUTHENTICATE z ustawionym ICV na same zera. Po pomyślnej weryfikacji, MAC polecenia EXTERNAL AUTHENTICATE staje się ICV dla późniejszych weryfikacji C/MAC i/lub generacji R-MAC.

Podczas używania jawnej inicjacji Bezpiecznego Kanału, jako ICV zostaje wykorzystany MAC obliczony na podstawie identyfikatora wybranej aplikacji (AID, ang. *Application Identifier*). Wartość ICV dla pierwszego obliczenia C-MAC nowej sesji Bezpiecznego Kanału oblicza się w następujący sposób:

- Zastosowanie odwrotnego wyrównywania dla identyfikatora AID wybranej aplikacji
- Obliczenie MAC z wykorzystaniem Single DES Plus Final Triple DES z kluczem CMAC, identyfikatora AID aplikacji oraz ICV składającym się z samych binarnych zer.

Powstały w ten sposób MAC zostaje ICV dla pierwszego C-MAC dla sesji Bezpiecznego Kanału. Wartość ICV dla pierwszego R-MAC jest obliczana w ten sam sposób, za wyjątkiem tego, że w drugim kroku do obliczenia MAC w miejscu klucza C-MAC używany jest klucz RMAC..

Szyfrowanie wektora

W ramach rozszerzenia mechanizmu C-MAC, ICV jest szyfrowany przed zastosowaniem go do obliczenia kolejnego C-MAC. Zastosowanym mechanizmem szyfrującym jest pojedynczy DES, z wykorzystaniem pierwszej połowy klucza sesji C-MAC. Pierwszy ICV sesji nie jest zaszyfrowany.

3.3.8 Klucze sesyjne

Klucze sesyjne DES są generowane za każdym razem, gdy inicjowany jest Bezpieczny Kanał i mogą być one używane do kolejnych poleceń, jeśli wymagana jest bezpieczna wymiana komunikatów. Są one generowane w celu zapewnienia, że dla każdej bezpiecznej sesji komunikacyjnej używany jest inny zestaw kluczy.

Klucze sesji DES są tworzone przy użyciu statycznego klucza Bezpiecznego Kanału, aktualnej wartości licznika sekwencji oraz wyrównania składającego się z binarnych zer. Tworzenie odbywa się w następujący sposób:

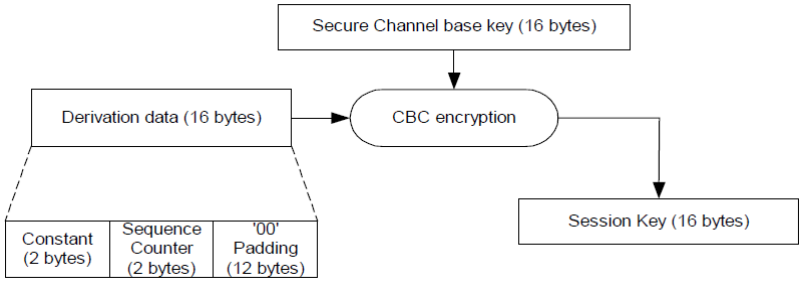
- Klucze sesyjne C-MAC (ang. *Secure Channel C-MAC session key*) – użycie klucza głównego lub klucza MAC (S-MAC) oraz danych dywersyfikacyjnych ze stałą '0101'

- Klucze sesyjne R-MAC (ang. *Secure Channel R-MAC session keys*) – użycie klucza głównego lub klucza MAC (S-MAC) oraz danych dywersyfikacyjnych ze stałą '0102'
- Klucze sesyjne szyfrujące (ang. *Secure Channel encryption session keys*) – użycie klucza głównego lub klucza szyfrującego (S-ENC) oraz danych dywersyfikacyjnych ze stałą '0182'
- Klucze sesyjne szyfrujące dane (ang. *Secure Channel data encryption session keys*) – użycie klucza głównego lub klucza szyfrowania danych (DEK) oraz danych dywersyfikacyjnych ze stałą '0181'

Tryb DES używany do generowania tych kluczy to zawsze potrójny DES w trybie CBC.

Klucze sesji R-MAC są generowane na podstawie aktualnej wartości licznika sekwencji w momencie inicjowania sesji RMAC. Jeżeli sesja R-MAC rozpoczyna się poleceniem EXTERNAL AUTHENTICATE, do wygenerowania klucza sesji R-MAC używana jest ta sama wartość licznika sekwencji, co do wygenerowania innych kluczy sesji dla tej samej Secure Channel Session.

Na rysunku 3.2 przedstawiono sposób generowania kluczy sesyjnych na podstawie klucza głównego:



Rysunek 3.2: Generowanie kluczy sesyjnych na podstawie klucza głównego [13]

3.3.9 Kryptogramy

W przypadku jawnej inicjacji Bezpiecznego Kanału, zarówno host jak i karta generują kryptogramy uwierzytelniania, które są następnie wzajemnie weryfikowane. Wygenerowanie lub weryfikacja kryptogramu uwierzytelniającego wykorzystuje klucz sesji S-ENC oraz metodę podpisywania z wykorzystaniem pełnego potrójnego DES (ang. *Full Triple DES*).

Kryptogram karty

Generowanie i weryfikacja kryptogramu karty odbywa się przez połączenie 8-bajtowego wyzwania hosta, 2-bajtowego licznika sekwencji i 6-bajtowego wyzwania karty, w wyniku czego powstaje 16-bajtowy blok. Dane podlegają wyrównaniu według wzorca: 80 00 00 00 00 00 00 00 00 00. Stosuje się metodę podpisu, używając klucza sesji S-ENC oraz ICV składającego się z binarnych zer. Do utworzonego 24 bajtowego bloku stosuje się metodę podpisu z wykorzystaniem klucza sesji S-ENC oraz ICV składającego się z binarnych zer. W rezultacie otrzymuje się 8-bajtową sygnaturę będącą kryptogramem karty.

Kryptogram hosta

Generowanie i weryfikacja kryptogramu hosta odbywa się przez połączenie 2-bajtowego licznika sekwencji, 6-bajtowego wyzwania karty oraz 8-bajtowego wyzwanie hosta, w wyniku czego powstaje 16-bajtowy blok. Dane podlegają wyrównaniu według wzorca: 80 00 00 00 00 00 00 00 00 00. Stosuje się metodę podpisu, używając klucza sesji S-ENC oraz ICV składającego się z binarnych zer. Do utworzonego 24 bajtowego bloku stosuje się metodę podpisu z wykorzystaniem klucza sesji S-ENC, ICV składającego się z binarnych zer oraz dodanych danych do wiadomości (padding). W rezultacie otrzymuje się 8-bajtową sygnaturę będącą kryptogramem hosta.

3.3.10 Wyzwanie karty

Jest to albo losowa albo pseudolosowa liczba unikalna dla bezpiecznej sesji. Pseudolosowe wyzwanie karty może zostać wygenerowane w następujący sposób:

- AID aplikacji żądającej otwarcia bezpiecznego kanału poddawany jest wyrównaniu
- MAC jest obliczany za pomocą Single DES Plus Final Triple DES MAC, przy użyciu klucza sesji C-MAC i ICV zer binarnych
- Sześć najdalej na lewo wysuniętych bajtów wynikowego MAC jest wyzwaniem karty.

Przy wykorzystaniu niejawnej inicjacji sesji, kryptogramem uwierzytelniania jest pierwszy CMAC otrzymany przez kartę od hosta.

3.3.11 C-MAC - tworzenie oraz weryfikacja

C-MAC jest generowany przez hosta i jest stosowany dla całego polecenia APDU przekazywanego do karty. Obejmuje nagłówek, pole danych, ale nie włącza maksymalnej długość danych oczekiwanych w odpowiedzi na polecenie (L_e).

Do generacji i weryfikacji C-MAC używany jest klucz sesyjny Secure Channel C-MAC oraz ICV z wykorzystaniem Single DES Plus Final Triple DES. ICV może być zaszyfrowany przed użyciem.

Dla każdego następnego polecenia następującego po pierwszej udanej weryfikacji C-MAC, ICV jest wartością C-MAC pomyślnie zweryfikowaną dla poprzedniego polecenia otrzymanego przez kartę.

Do utworzonego 24 bajtowego bloku stosuje się metodę podpisu z wykorzystaniem klucza sesji C-MAC, ICV składającego się z binarnych zer oraz wartości wynikającej ze stosowanego wzorca wyrównania do rozmiaru bloku (padding). W rezultacie otrzymuje się 8-bajtową sygnaturę będącą kryptogramem hosta.

W celu uwzględnienia obecności C-MAC w komunikacie poleceń, nagłówek polecenia jest modyfikowany w następujący sposób:

- Długość komunikatu (L_c) jest zwiększana o 8, w celu określenie, że C-MAC będzie uwzględniony w polu komunikatu,

- Bajt klasy jest modyfikowany w celu określenia, że to polecenie APDU obejmuje bezpieczne przesyłanie wiadomości.

C-MAC jest dodawany na końcu komunikatu polecenia APDU z wyłączeniem Le, ale z uwzględnieniem modyfikacji dokonanych w nagłówku polecenia (klasa i L_c).

Do utworzenia C-MAC zdefiniowano dwie następujące metody:

- Generowanie C-MAC na niemodyfikowanym APDU: wypełnienie polecenia APDU jest wymagane przed wykonaniem generacji C-MAC, a modyfikacja nagłówka polecenia APDU jest wymagana po wykonaniu generacji C-MAC;
- Generowanie C-MAC na zmodyfikowanym APDU: wypełnienie polecenia APDU i modyfikacja nagłówka polecenia jest wymagana przed wykonaniem generacji C-MAC.

Jeżeli nie jest wymagana żadna inna bezpieczna wymiana komunikatów, otrzymany komunikat jest już przygotowywany do transmisji na kartę.

Karta, w celu weryfikacji C-MAC, wykorzystuje ten sam mechanizm wypełniania danych, używa tego samego klucza sesji ICV i C-MAC, które używał host w celu weryfikacji C-MAC. Zweryfikowany C-MAC jest zachowywany i wykorzystywany jako ICV dla każdej kolejnej weryfikacji C-MAC. Jest to prawdą niezależnie od tego, czy APDU zostało pomyślnie zakończone, tj. zweryfikowany C-MAC nigdy nie może zostać odrzucony na rzecz wcześniej zweryfikowanej wartości C-MAC.

3.3.12 Polecenia APDU

INITIALIZE UPDATE

Polecenie INITIALIZE UPDATE jest używane, podczas jawnej inicjacji Bezpiecznego Kanału, do przesyłania danych o karcie i sesji pomiędzy kartą a hostem. To polecenie inicjuje sesji Bezpiecznego Kanału. W każdej chwili podczas trwania bezpiecznego kanału, polecenie INITIALIZE UPDATE może zostać wydane karcie w celu zainicjowania nowej sesji.

Budowa polecenie została przedstawiona w poniższej tabeli:

Kod	Wartość	Opis
CLA	80 – 83 lub C0 - CF	
INS	50	INITIALIZE UPDATE
P1	XX	Numer wersji klucza
P2	00	Referencyjny parametr kontrolny P2
L_c	08	Długość wyzwania hosta
Data	XX XX	Wyzwanie hosta
L_e	00	

Numer wersji klucza definiuje numer wersji klucza w obrębie Domeny Bezpieczeństwa, który ma być użyty do zainicjowania sesji. Jeśli wartość ta wynosi zero, zostanie użyty pierwszy dostępny

klucz wybrany przez domenę bezpieczeństwa. Referencyjny parametr kontrolny P2 ustawia się zawsze na "00". Pole danych zawiera 8 bajtowe wyzwania hosta. Powinno być unikalne dla danej sesji. Pole danych komunikatu zwrotnego zawiera następujące elementy:

Nazwa	Długość [bajty]
Dane dywersyfikacyjne	10
Informacja o kluczu	2
Licznik sekwencji	2
Wyzwanie karty	6
Kryptogram karty	8

Dane dotyczące dywersyfikacji kluczy to dane wykorzystywane w procesie dywersyfikacji. Informacje o kluczu obejmują numer wersji klucza i Protokołu Bezpiecznego Kanału używane przy inicjowaniu sesji. Dla SCP02 wartość wynosi '02'. Licznik sekwencji jest wewnętrznym licznikiem przyrostowym używanym do tworzenia kluczy sesji. Wyzwanie związane z kartą jest wewnętrznie generowaną liczbą losową. Kryptogram karty jest kryptogramem uwierzytelniania. Pomyślne wykonanie polecenia powinno być potwierdzone bajtami statusu '90' '00'

EXTERNAL AUTHENTICATE

Polecenie EXTERNAL AUTHENTICATE jest używane przez kartę podczas jawnej inicjacji Bezpiecznego Kanału. Wykorzystywane jest do uwierzytelniania hosta i określenia poziomu bezpieczeństwa wymaganego dla wszystkich kolejnych poleceń. Przed wywołaniem tego polecenia wymagane jest pomyślne wykonanie polecenia INITIALIZE UPDATE.

Budowa polecenie została przedstawiona w poniższej tabeli:

Kod	Wartość	Opis
CLA	84 – 84 lub E0 – EF	
INS	82	EXTERNAL AUTHENTICATE
P1	XX	Poziom bezpieczeństwa
P2	00	Referencyjny parametr kontrolny P2
L_c	08	Długość kryptogramu hosta oraz MAC
Data	XX XX	Kryptogram hosta oraz MAC
L_e		Nieobecne

Rozdział 4

Przegląd istniejących rozwiązań

Na rynku oprogramowania istnieje wiele rozwiązań pozwalających na bezpieczne tworzenie, przechowywanie oraz wykorzystywanie kluczy kryptograficznych. Celem tego rozdziału jest porównanie dostępnych urządzeń i ich charakterystyk. Pozwoli to na wybranie odpowiedniego modelu, który będzie mógł pracować jako Hardware Security Module.

4.1 Karty elektroniczne

Aby zwiększyć bezpieczeństwo klucza prywatnego, możne być on przechowywany na urządzeniu, takim jak karta elektroniczna. Dostęp do klucza prywatnego w takim przypadku, wymaga dostępu do karty elektronicznej oraz znajomości kodu PIN.

4.1.1 Elektroniczna Legitymacja Studencka

Elektroniczna Legitymacja Studencka (*ELS*) [36] jest elektroniczną kartą procesorową wyposażoną w dwa interfejsy: stykowy zgodny z normą ISO/IEC 7816 o pojemności pamięci 80kB oraz bezstykowy określony normą ISO/IEC 14443 typ A o pojemności pamięci 1kB. Specyfikacja karty, jej wygląd oraz sposób wykonania są zgodne z Rozporządzeniem Ministra Nauki i Szkolnictwa Wyższego z dnia 27 września 2018 r. w sprawie studiów [38]. Karta jest zgodna z Java Card 2.2.1 [6] oraz GlobalPlatform Card Specification 2.1.1 [12]. Według specyfikacji, karta posiada wbudowany koprocesor kryptograficzny oraz posiada możliwość generowania par kluczy dla RSA do 2048 bitów i ECC.

4.1.2 SmartCard-HSM

SmartCard-HSM [33] jest lekkim sprzętowym modułem bezpieczeństwa w postaci karty elektronicznej, MicroSD lub USB. Zapewnia zdalnie zarządzalny, bezpieczny magazyn kluczy do ochrony kluczy RSA i ECC. SmartCard-HSM[40] jest dostępny w standardowych rozmiarach karty kredytowej (ID–1), karty SIM (ID–000) lub nawet w specjalnych formatach, takich jak ID–00. Działa na stykowym i/lub bezstykowym interfejsie ze standardowymi czytnikami, które działają zgodnie z normami: ISO–7816 lub ISO–14443. Współpracuje on z systemami Windows, Linux i Mac. SmartCard–HSM 4K umożliwia zarządzanie kluczami RSA do 4096 bitów, kluczami ECC do 521 bitów i kluczami AES o długości 128, 192 i 256 bitów. Maksymalna liczba kluczy oznacza liczbę kluczy prywatnych, które można utworzyć na pustym urządzeniu. Liczba ta może być mniejsza w przypadku przechowywania dodatkowych metadanych lub certyfikatów. Urządzenie posiada 76KB EEPROM, pozwala na przechowywanie do 19 kluczy RSA 4096 lub 38 kluczy RSA 2048 lub 30 kluczy ECC 521 lub 304 kluczy AES 256. Cena urządzenia wynosi 34€ [23].

4.1.3 OpenPGP Card

OpenPGP Card [25] jest kartą elektroniczną zgodną ze specyfikacją ISO/IEC 7816–4/–8. Jest ona zintegrowana z wieloma funkcjami GnuPG. Karta występuje w postaci standardowego rozmiaru ID–1 oraz ID–1 z wycięciami dla ID–000. Karta w specyfikacji V3.3 pozwala na przechowywanie kluczy RSA od 2048 do 4096 bitów oraz ECC od 256 do 521 bitów, zgodnych z parametrami organizacji standaryzacyjnych NIST/ANSI oraz Brainpool. Cena karty w specyfikacji V3.3 wynosi około 18€.

4.1.4 PIVKey

Urządzenia z serii PIVKey C900 [15] to karty elektroniczne o wielkości ID-1. PIVKey C980 Enterprise Dual PKI Smart Card obsługuje czytniki stykowe (ISO–7816) oraz zbliżeniowe (ISO–14443), umożliwiając bezpieczne przechowywanie i korzystanie z certyfikatów cyfrowych X.509 oraz związanych z nimi kluczy kryptograficznych. PIVKey jest dostarczany z jednym certyfikatem urządzenia do testowania i do prostych zastosowań. Dodatkowe certyfikaty mogą być załadowane do PIVKey za pomocą PIVKey Windows Minidriver and Admin tools, które są dostępne bezpłatnie na stronie PIVKey.com. Urządzenie może być używane na dowolnym standardowym komputerze PC z systemem Windows przy użyciu sterownika In-box PIV. Może być używany z Urzędami Certyfikacji (*CA*), a certyfikaty mogą być importowane za pomocą windowsowego narzędzia „certutil". Na Mac oraz Linux wykorzystanie karty jest możliwe przy użyciu dowolnego oprogramowania pośredniczącego zgodnego z PIV. Wykorzystanie middleware, np. OpenSC powoduje, że karta działa jedynie w trybie odczytu. Urządzenie umożliwia przechowywanie kluczy RSA 1024 oraz 2048 bitów. Chip bezpieczeństwa i system operacyjny są zgodne z FIPS 140-2, poziom 3. Cena karty wynosi 19$.

4.1.5 ACOS5–64 v3.00

ACOS5–64 v3.00 [21] jest kartą kryptograficzną zgodną ze standardem FIPS 140-2 Level 3 oraz normami: ISO7816–1/2/3/4/8/9, Common Criteria EAL5+. Oferuje rozwiązanie programowe PKI, które zawiera aplikacje i oprogramowanie pośredniczące. Umożliwia to używanie jej na platformach Windows, Linux i MAC. Karta posiada 64KB pamięci EEPROM. Wspierane są następujemy algorytmy kryptograficzne: DES, 3DES, 3K3DES, AES–128, AES–192, AES–256 oraz RSA do 4096 bitów. Urządzenie jest zgodne z Microsoft Crypto–API, Microsoft CNG oraz oprogramowaniem pośredniczącym dla PKCS#11. Karty nie są obsługiwane przez OpenSC, jednak producent dostarcza niezbędne sterowniki PC/SC do zarządzania urządzeniem. Karta dostępna jest w cenie 10$. Istnieje również możliwość zakupu pakietu ACOS5-64 Client Kit. W jego skład wchodzi 5 kart elektronicznych oraz dwa rodzaje pakietów, dla administratora oraz użytkownika. Pakiet administratora przeznaczony jest dla urzędów certyfikujących i rejestracyjnych, także dla administratorów, którzy planują wykorzystać kartę elektroniczną do użytku w PKI. Z kolei pakiet użytkownika jest przeznaczony dla użytkowników końcowych, którzy będą wykorzystać kartę w aplikacjach powiązanych z PKI, wymagających certyfikatów cyfrowych. Cena zestawu wynosi 66$.

4.1.6 Aventra MyEID PKI

Aventra MyEID PKI Card [1] jest kartą JavaCard z apletem Aventra MyEID. Może być używana z systemami operacyjnych Windows, Linux oraz Mac OS. Karta współpracuje między innymi z Windows Smart Card Minidriver (Windows certified) oraz sterownikiem MyClient obsługującym CSP oraz PKCS#11. Karta posiada 80KB EEPROM, może przechowywać klucze RSA oraz ECC od 256 do 521 bit. Wspiera szyfrowanie symetryczne z wykorzystaniem AES oraz 3DES. Cena karty wynosi 11€.

4.1.7 Oberthur ID-One Cosmo

Oberthur ID-One Cosmo v7 [31] to karta elektroniczna zgodna ze specyfikacją Java Card 2.2.2 oraz GlobalPlatform 2.1.1. Urządzenie posiada certyfikat Common Criteria EAL+ oraz jest zgodne z FIPS 140–2 Level 3. Obsługuje mechanizmy kryptograficzne, takie jak 3DES, AES, RSA do 2048 bitów oraz krzywe eliptyczne do 521 bitów. Występuje ona w wersjach stykowych, zbliżeniowych oraz dualnych. Ilość pamięci w zależności od wersji wynosi od 16 do 128KB. Karta w wersji v7.0 jest zgodna z normą ISO–7816 oraz ISO–14443. Pozwala na łatwą integrację z Microsoft CSP oraz oprogramowaniem pośredniczącym PKCS#11.

4.1.8 Tokeny USB

Token USB [39] to jedno urządzenie, w ramach którego otrzymujemy czytnik kart połączony z chipem karty elektronicznej. Dzięki temu urządzenia te nie wymagają dodatkowego sprzętu i działają po podłączeniu do portu USB w komputerze.

4.1.9 SmartCard–HSM

SmartCard–HSM [40] jest dostępny w wersji USB-Token, która jest czytnikiem kart zgodym z CCID połączonym z chipem karty w jednym urządzeniu. USB-Token nie wymaga dodatkowego sprzętu ani sterownika i działa natychmiast po podłączeniu do portu USB w komputerze. Wersja EA+ zawiera najnowszą wersję apletu. Pozwala ona na na uwierzytelnianie na podstawie klucza publicznego, obsługę PIN, ograniczenia kluczy zdefiniowane przez użytkownika oraz możliwość ograniczenia licznika wykorzystania klucza. Urządzenie pozwala na przechowywanie do 32 kluczy RSA 2048 lub 40 kluczy ECC 256 . SmartCard–HSM 4K umożliwia zarządzanie kluczami RSA do 4096 bitów, kluczami ECC do 521 bitów i kluczami AES o długości 128, 192 i 256 bitów. Maksymalna liczba kluczy oznacza liczbę kluczy prywatnych, które można utworzyć na pustym urządzeniu. Urządzenie tak samo jak karta elektroniczna, posiada 76KB EEPROM. Pozwala na przechowywanie do 19 kluczy RSA 4096 lub 38 kluczy RSA 2048 lub 30 kluczy ECC 521 lub 304 klucze AES 256. Cena urządzenia SmartCard–HSM 4K USB-Token wynosi 59€ [22].

4.1.10 YubiHSM

Możliwości YubiHSM 2 [35] obejmują generowanie oraz zapisywanie kluczy, podpisywanie, deszyfrowanie, haszowanie oraz kodowanie. Urządzenie gwarantuje szerokie możliwości kryptograficzne: RSA, ECC, ECDSA (ed25519), SHA-2, AES. Przechowuje do 127 kluczy RSA 2048, 93 RSA 3072, 68 RSA 4096 lub 255 ECC, przy założeniu obecności tylko jednego klucza uwierzytelniania. Funkcje YubiHSM 2 są dostępne poprzez integrację z otwartym oprogramowaniem oraz kompleksowym pakietem narzędzi programistycznych (*SDK*). Obsługuje następujące interfejsy kryptograficzne: Microsoft CNG (KSP), PKCS#11, natywne biblioteki YubiHSM (C, python). Pozwala na jednoczesne nawiązanie do 16 równoległych połączeń. Wiele aplikacji może tworzyć sesje z YubiHSM do wykonywania operacji kryptograficznych. Sesje mogą być automatycznie przerywane po nieaktywności lub mogą być długotrwałe w celu poprawy wydajności poprzez wyeliminowanie czasu tworzenia sesji. YubiHSM przechowuje wewnętrznie dziennik wszystkich zdarzeń związanych z zarządzaniem i operacjami kryptograficznymi, które występują w urządzeniu. Format urządzenia pozwala na umieszczenie HSM całkowicie wewnątrz portu USB-A. Pozwala to na ukrycie urządzenia, dzięki temu nie ma żadnych zewnętrznych części, które wystają z tylnej lub przedniej obudowy serwera. Cena produktu wynosi 650$.

4.2 HSM

Hardware Security Module (*HSM*) [19] to dedykowany procesor kryptograficzny, zgodny z normami FIPS 140–2 oraz Common Criteria wykorzystywany jako urządzenie do ochrony cyklu życia kluczy kryptograficznych. HSM jako odporne na włamania urządzenia doskonale sprawdzają się w zabezpieczaniu kluczy kryptograficznych oraz dostarczaniu usług szyfrowania, deszyfrowania, uwierzytelniania oraz podpisywania cyfrowego dla szerokiej gamy aplikacji. Głównymi dostawcami tego typu urządzeń są Thales, Gemalto oraz Utimaco. Ze względu na wysokie koszty takich urządzeń, które wynoszą ponad 30000$ [27], nie były one brane pod uwagę w wyborze preferowanego rozwiązania do testowania.

Rozdział 5

Karta elektroniczna jako HSM

5.1 Wstęp

W procesie zarządzania cyklem życia kluczy kryptograficznych bardzo ważną rolę odgrywają narzędzia oraz procesy pozwalające zagwarantować odpowiedni poziom bezpieczeństwa w celu zapobiegania nieautoryzowanemu dostępowi do danych poufnych. Z punktu widzenia bezpieczeństwa bardzo ważne jest, aby tylko odpowiednie osoby posiadały dostęp do tak krytycznych danych, ponieważ dostęp nieupoważnionych osób może prowadzić do utraty istotnych informacji. Jedynym z urządzeń, które może zostać wykorzystane do zapewnienia dodatkowej warstwy ochrony jest sprzętowy moduł bezpieczeństwa. Pozwala on chronić klucze na każdym etapie ich cyklu życia, w tym podczas tworzenia, importu, eksportu, wykorzystania, rotacji i niszczenia.

W ramach pracy zbadano możliwości przechowywania kluczy kryptograficznych na urządzeniu fizycznym oraz rozwiązaniu programowym. Przygotowano aplikację do zabezpieczenia oraz zarządzania kluczami kryptograficznymi znajdującymi się na karcie elektronicznej. Zaproponowane rozwiązanie pozwala na wykorzystanie ich w procesie wymiany kluczy dla Domeny Bezpieczeństwa na kartach elektronicznych przygotowywanych przez Międzyuczelniane Centrum Personalizacji Legitymacji Studenckiej. Do przygotowania rozwiązania w postaci aplikacji konsolowej napisanej w języku C#, wykorzystano środowisko programistyczne Microsoft Visual Studio 2019, które umożliwia łatwe zarządzanie kodem oraz pakietami NuGet.

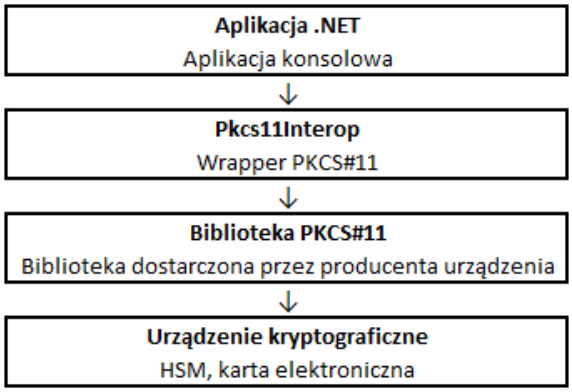
Jako urządzenie pozwalające na współpracę z kartami elektronicznymi, użyto czytnika GemPC Twin [41], który obsługuje wszystkie mikroprocesorowe karty elektroniczne ISO 7816. Czytnik jest dostarczany w komplecie ze sterownikami systemu Windows zatwierdzonymi przez WHQL, a także obsługuje inne systemy operacyjne, takie jak Linux.

5.2 Biblioteki

W pracy zostały wykorzystane biblioteki Pkcs11Interop oraz PCSC. Umożliwiają one przeprowadzenie operacji wymaganych do prawidłowego zarządzania cyklem życia kluczy oraz wykonywania operacji z ich udziałem.

5.2.1 Pkcs11Interop

Pkcs11Interop [17] jest otwartą i bezpłatną do użytku komercyjnego biblioteką, zgodną ze specyfikacją PKCS#11 v2.40 oraz schematem PKCS#11 URI. Pozwala na przeniesienie funkcjonalności PKCS#11 do środowiska .NET. Jest obsługiwana w systemach Windows, Linux, Mac OS X, Android i iOS, zarówno na platformach 32-bitowych, jak i 64-bitowych. W przygotowanym rozwiązaniu, biblioteka jest wykorzystywana w procesie zarządzania cyklem życia kluczy znajdujących się na karcie elektronicznej oraz do wykonywania operacji kryptograficznych wymaganych w czasie trwania procesu wymiany kluczy dla Domeny Bezpieczeństwa. Rysunek 5.1 przedstawia typowe wykorzystanie biblioteki Pkcs11Interop w aplikacji .NET.



Rysunek 5.1: Użycie biblioteki Pkcs11Interop w aplikacji .NET

5.2.2 pcsc-sharp

Biblioteka pcsc-sharp [7] zapewnia dostęp do Personal Computer/Smart Card Resource Manager dzięki wykorzystaniu PS/SC API. Implementuje częściową obsługę ISO/IEC 7816 i jest napisana tak, aby działała zarówno w systemie Windows, jak i Linux. Karty elektroniczne pracują z zestawem poleceń wysyłanych z aplikacji na kartę w bloku znanym jako APDU (Application Protocol Data Unit), który jest jednostką komunikacyjną pomiędzy czytnikiem a kartą. Wykorzystanie tej biblioteki pozwoliło na wysyłanie poleceń APDU do karty docelowej oraz odbieranie odpowiedzi przez hosta.

5.3 Oprogramowanie

W pracy zostały wykorzystane aplikacje SoftHSM oraz Java Card Developement Kit (JCKit), które umożliwiły programową emulację zachowania kart elektronicznych. Dzięki temu można było sprawdzić zachowanie urządzeń w przypadku wywoływania różnych funkcji oraz metod bez obaw o uszkodzenie fizycznego urządzenia w wyniku nieudanych prób uwierzytelniania lub wykonania niedozwolonych operacji. W przypadku wystąpienia błędów, programowe rozwiązanie można bezproblemowo przywrócić do stanu domyślnego i ponownie użyć do dalszych badań.

5.3.1 SoftHSM

Jednym z narzędzi była aplikacja SoftHSM, która jest programową implementacją urządzenia kryptograficznego dostępnego poprzez interfejs PKCS#11 i jest rozwijana jako część projektu OpenDNSSEC. Tego rozwiązania można używać do testowania funkcjonalności PKCS#11 bez konieczności posiadania sprzętowego modułu bezpieczeństwa. W celu wykonywania operacji kryptograficznych, można wykorzystać biblioteki kryptograficzne Botan lub OpenSSL [9]. Oprogramowanie nie implementuje wszystkich możliwości karty elektronicznej. Zostało wykorzystane w celu potwierdzenia możliwości użycia bibliotek PKCS#11 do zarządzania cyklem życia kluczy oraz przeprowadzenia wymaganych operacji kryptograficznych. Zarówno karta, jak i SoftHSM korzystają z tych samych klas, dlatego kod programu dla obydwu rozwiązań jest taki sam. Różnią się tylko biblioteką dostarczaną przez producenta. W przypadku przygotowywanego rozwiązania nie było potrzeby użycia żadnych dodatkowych funkcjonalności, które oferuje karta, zatem wybrane oprogramowanie okazało się wystarczające. Emulator pełnej funkcjonalności karty elektronicznej byłby bardziej odpowiedni do użycia w przypadku wyboru opcji stworzenia apletu, który miałby zostać zainstalowany na karcie.

W czasie przygotowywania rozwiązania oraz fazy testów wykorzystywany był SoftHSM w wersji 2.5.0. Pakiet instalacyjny SoftHSM2 installer for MS Windows pozwolił na zainstalowanie oprogramowania w środowisku Windows. Pakiet ten zawierał zarówno 32-bitowe, jak i 64-bitowe wersje biblioteki PKCS#11.

W celu rozpoczęcia używania SoftHSM było konieczne przeprowadzenie inicjalizacji. W wyniku tej operacji utworzono nowy token o etykiecie Token 1 i określony PIN dla administratora oraz dla użytkownika:

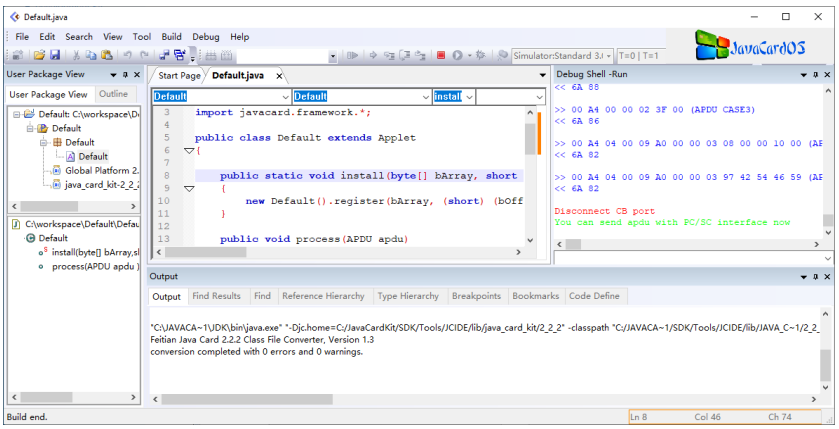
```
$ softhsm2-util.exe --init-token --slot 0 --label "Token 1"
=== SO PIN (4-255 characters) ===
Please enter SO PIN: ********
Please reenter SO PIN: ********
=== User PIN (4-255 characters) ===
Please enter user PIN: ****
Please reenter user PIN: ****
The token has been initialized and is reassigned to slot 1507644569
```

Prawidłowe wykonanie polecenia zostało potwierdzone poprzez sprawdzenie listy dostępnych slotów na urządzeniu:

```
$ softhsm2-util --show slots
Available slots:
Slot 1507644569
    Slot info:
        Description:      SoftHSM slot ID 0x59dcd499
        Manufacturer ID:  SoftHSM project
        Hardware version: 2.5
        Firmware version: 2.5
        Token present:    yes
    Token info:
        Manufacturer ID:  SoftHSM project
        Model:            SoftHSM v2
        Hardware version: 2.5
        Firmware version: 2.5
        Serial number:    8643f700d9dcd499
        Initialized:      yes
        User PIN init.:   yes
        Label:            Token 1
```

5.3.2 Java Card Developement Kit (JCKit)

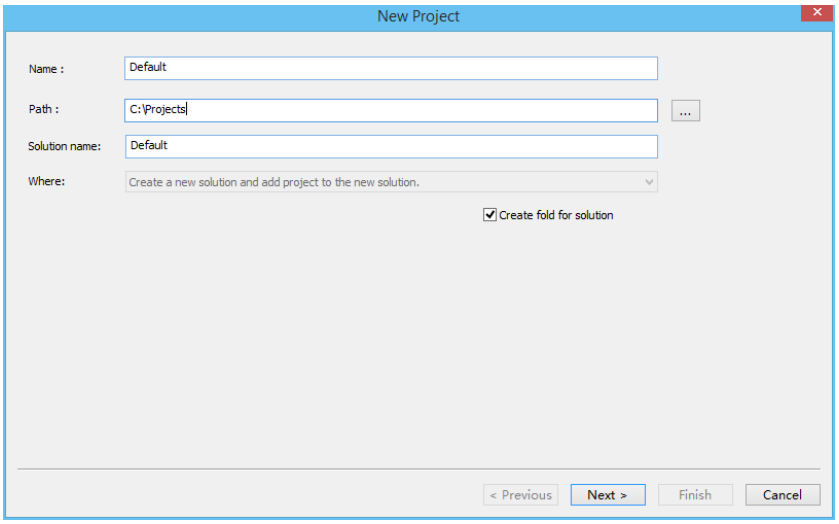
Java Card Developement Kit (JCKit) [29] zawierający JCIDE i PyApdutool, zapewnia kompletne środowisko dla programistów kart elektronicznych. JCIDE umożliwia podłączenie dowolnej aplikacji zgodnej z PC/SC i otrzymanie odpowiedzi za pomocą zintegrowanego wirtualnego interfejsu PC/SC. Takie rozwiązanie pozwala potwierdzić, że tworzona aplikacja wysyła oraz odbiera prawidłowe odpowiedzi dla wywoływanych poleceń APDU. Wirtualny czytnik kart (ang. *Virtual Contact Reader*) jest jedną z funkcji w JCIDE. Pozwala on symulować kartę lub czytnik w oparciu o protokół komunikacyjny PCSC w systemie Windows, który może być używany niemal tak samo jak fizyczne urządzenie. Po instalacji w systemie pojawiły się następujące czytniki kart wirtualnych: JAVACOS Virtual Contact Reader 0 oraz JAVACOS Virtual Contactless Reader 1. Debug shell umożliwił śledzenie poleceń wymienianych pomiędzy hostem oraz kartą w celu identyfikacji błędów podczas procesu tworzenia oraz testowania przygotowywanego rozwiązania. Na rysunku 5.2 przedstawiono interfejs programu po uruchomieniu projektu.



Rysunek 5.2: Interfejs graficzny JCIDE

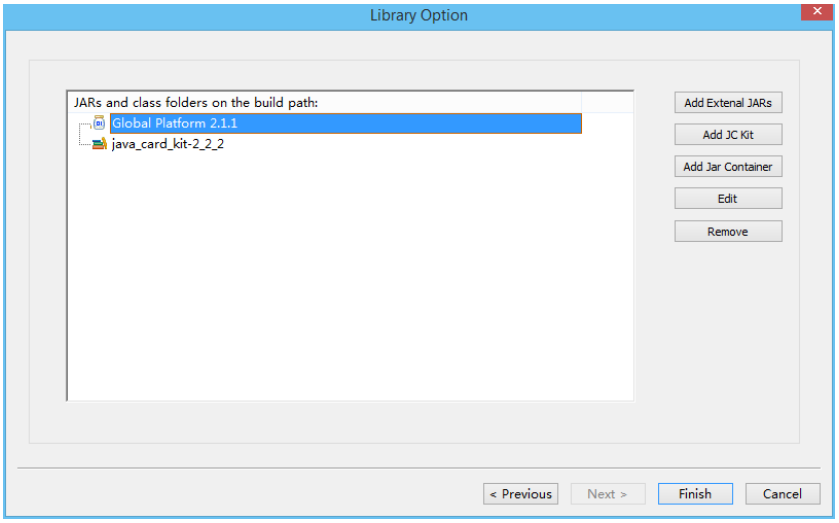
Konfiguracja JCIDE

Przed rozpoczęciem korzystania z emulatora karty obsługującej protokół SCP02, należało utworzyć nowy projekt, a następnie pakiet oraz aplet. W celu uruchomienia kreatora tworzenia projektu, z górnego paska w aplikacji wybrano opcję File → New → Project. Następnie zdefiniowano nazwę projektu oraz wybrano lokalizację, w której znajdowały się pliki związane z tym rozwiązaniem (rys. 5.3).



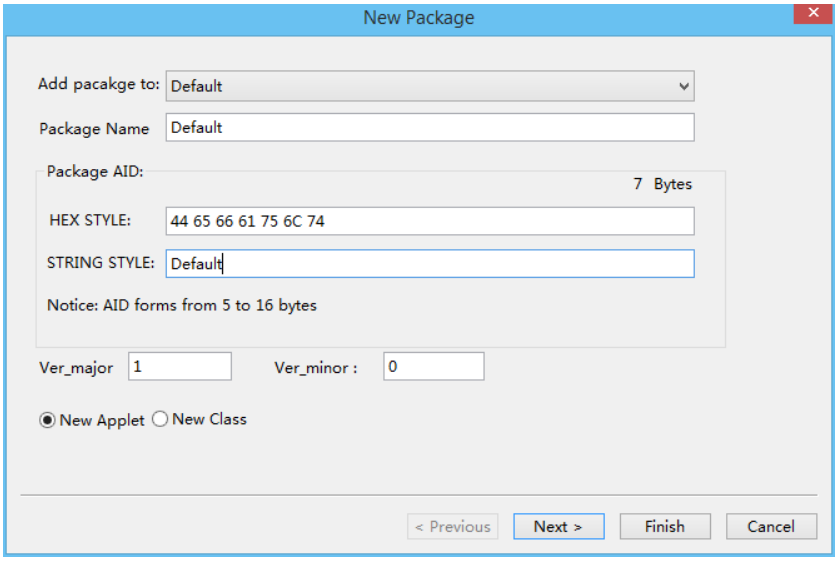
Rysunek 5.3: Definiowanie nazwy nowego projektu oraz lokalizacji

Tworzony projekt działał w oparciu o specyfikację Global Platform w wersji 2.1.1 oraz wykorzystywał Java Card Kit w wersji 2.2.2 (rys. 5.4).



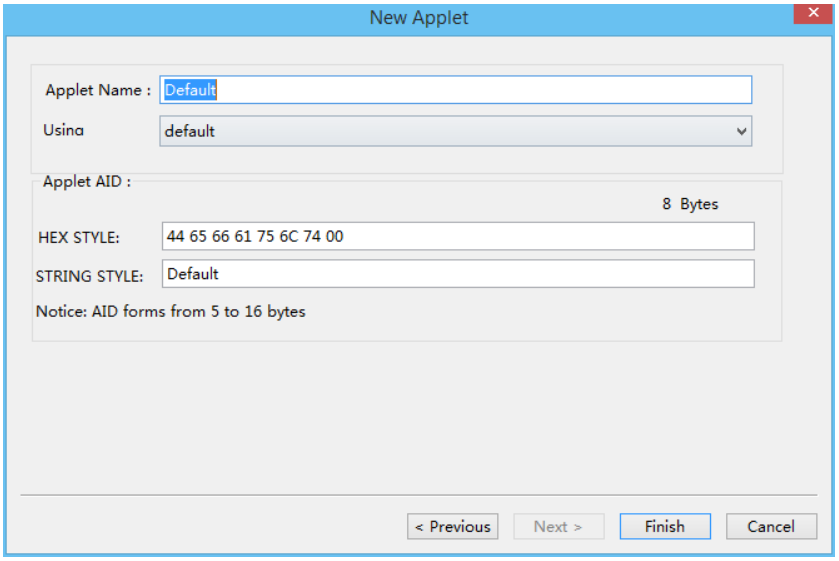
Rysunek 5.4: Określenie wykorzystywanych bibliotek

Do poprawnego funkcjonowania, należało określić nazwę dla tworzonego pakietu oraz jego AID (rys. 5.5).



Rysunek 5.5: Nadanie nazwy nowego pakietu

Na koniec wybrano nazwę apletu oraz ustalono jego AID (rys. 5.6).

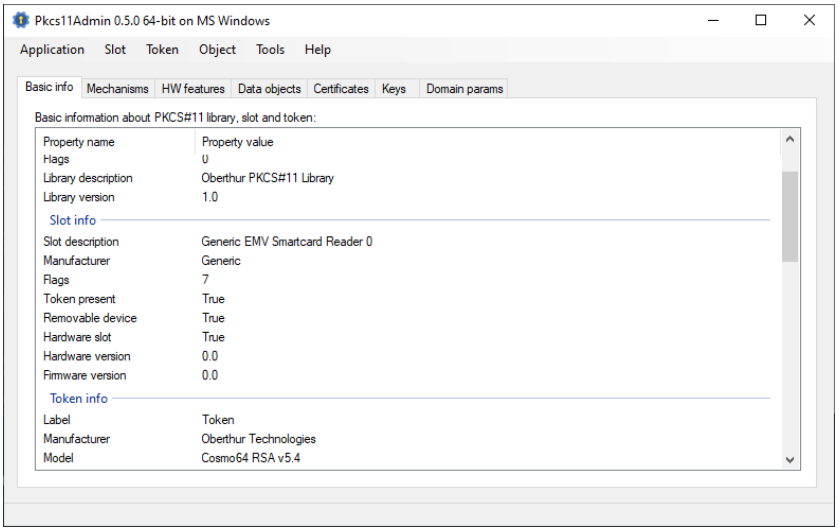


Rysunek 5.6: Ustalanie nazwy nowego apletu

Tak przeprowadzona konfiguracja, pozwoliła na uruchomienie projektu, który symulował działanie karty elektronicznej wykorzystywanej w dalszych etapach pracy.

5.4 Pkcs11Admin

Pkcs11Admin [16] to narzędzie do administrowania urządzeniami z obsługą PKCS#11.Pozwala między innymi na zarządzanie kartami elektronicznymi oraz sprzętowymi modułami bezpieczeństwa. Narzędzie działa na systemach operacyjnych Windows, Linux i Mac OS X. Umożliwia sprawdzenie stanu urządzenia, jego możliwości, posiada obsługę zarządzania tokenami oraz kodami PIN. Zarządza poszczególnymi atrybutami obiektów, importuje oraz eksportuje je. Umożliwia generowanie kluczy oraz par kluczy. Posiada możliwość tworzenia żądań podpisania certyfikatu oraz przechowywania certyfikatów X.509. Do prawidłowego działania narzędzie wymaga zewnętrznej biblioteki PKCS#11 dostarczonej przez producenta urządzenia. Na rysunku 5.7 przedstawiono interfejs graficzny narzędzia.



Rysunek 5.7: Interfejs graficzny Pkcs11Admin

5.5 GlobalPlatformPro

Przed rozpoczęciem testów przygotowanego rozwiązania, sprawdzono możliwość przeprowadzania operacji wymiany klucza przy użyciu jednego z rozwiązań dostępnych na rynku. Do tego celu wykorzystano GlobalPlatformPro [8], narzędzie wiersza poleceń typu open source umożliwiające zarządzania apletami dla kart elektronicznych. Najnowsza wersja GlobalPlatformPro została pobrana z oficjalnego repozytorium GitHub.

5.5.1 Zmiana klucza

Zgodnie z instrukcją znajdującą się w repozytorium wykonano operację blokowania karty za pomocą klucza 010B0371D78377B801F2D62AFC671D95. W przypadku testów, zarówno karta jak i rozwiązanie programowe posiadały początkowo domyślny klucz karty 40414243444546474849 4A4B4C4D4E4F, dlatego jego wartość nie była przekazywana w ramach pierwszego polecenia. Narzędzie było uruchamiane w trybie z wypisywaniem szczegółów, co pozwoliło zaobserwować poszczególne kroki w procesie ustawiania nowych kluczy. Dzięki temu w trakcie działania aplikacji można sprawdzić wysyłane polecenia APDU do karty, zwracane odpowiedzi oraz wartości zmiennych dla poszczególnych kroków obliczeń.

5.5.2 JCIDE

W pierwszej kolejności testy przeprowadzono na rozwiązaniach programowych, żeby poznać zasadę działania aplikacji oraz uniknąć ryzyka zablokowania fizycznej karty elektronicznej w wyniku wywoływania nieprawidłowych poleceń wiersza poleceń.

Przeprowadzenie operacji zmiany klucza karty możliwe było dzięki uruchomieniu programu z opcją –lock oraz podaniem nowej wartości klucza.

```
.\gp.exe -d --lock 010B0371D78377B801F2D62AFC671D95 --reader "JAVACOS Virtual
↪  Contact Reader 0"
...
A>> T=1 (4+0000) 00A40400 00
A<< (0094+2) (11ms) 6F5C8408A0000000030000000A550734A06072A864886FC6B016
00C060A2A864886FC6B02020101630906072A864886FC6B03640B06092A8648
86FC6B040215650B06092B851086486402010366 0C060A2B060104012A026E01029F6501FF
9000
...
Warning: no keys given, using default test key 404142434445464748494A4B4C4D4E4F
...
[TRACE] GPSession - Generated host challenge: 8A7C02D6AFF12B5B
A>> T=1 (4+0008) 80500000 08 8A7C02D6AFF12B5B 00
A<< (0028+2) (6ms) 00000000000000000000010200003D029C31C7899C6F631B147B3E1A 9000
```

```
[DEBUG] GPSession - Host challenge: 8A7C02D6AFF12B5B
[DEBUG] GPSession - Card challenge: 00003D029C31C789
[DEBUG] GPSession - Card reports SCP02 with key version 1 (0x01)
...
[INFO] GPSession - Session keys:
ENC=010B0371D78377B801F2D62AFC671D95
MAC=D1C28C601652A4770D67AD82D2D2E1C4
RMAC=FFAEC7EC7FAD69F9FBFF093BF2F79C45,
...
[DEBUG] GPSession - Verified card cryptogram: 9C6F631B147B3E1A
[DEBUG] GPSession - Calculated host cryptogram: 154A72DBD0BC5F1E
[DEBUG] SCP02Wrapper - MAC input: 8482010010154A72DBD0BC5F1E
A>> T=1 (4+0016) 84820100 10 154A72DBD0BC5F1EE111AF9A8C97B747
A<< (0000+2) (6ms) 9000
[DEBUG] SCP02Wrapper - MAC input: 84CA00E008
A>> T=1 (4+0008) 84CA00E0 08 D0700E7D427F3278 00
A<< (0020+2) (8ms) E012C00401018010C00402018010C00403018010 9000
...
[DEBUG] GPSession - PUT KEY version
ENC=010B0371D78377B801F2D62AFC671D95 (KCV: F2DCDD)
↪  MAC=010B0371D78377B801F2D62AFC671D95 (KCV: F2DCDD)
↪  DEK=010B0371D78377B801F2D62AFC671D95 (KCV: F2DCDD) for null
[DEBUG] SCP02Wrapper - MAC input: 84CA00E008
A>> T=1 (4+0008) 84CA00E0 08 7DA7E0ED3C1D52A9 00
A<< (0020+2) (9ms) E012C00401018010C00402018010C00403018010 9000
...
[DEBUG] PlaintextKeys - Encrypting ENC value 010B0371D78377B801F2D62AFC671D95
↪  with 404142434445464748494A4B4C4D4E4F
[DEBUG] PlaintextKeys - Encrypting ENC value 010B0371D78377B801F2D62AFC671D95
↪  with E11987EE331B417A5D67D760692F89D4
...
[DEBUG] SCP02Wrapper - MAC input:
↪  84D801814B0180104B5D0DA613A894CF68ADDD849A2F63FE03F2DCDD
80104B5D0DA613A894CF68ADDD849A2F63FE03F2DCDD80104B5D0DA6
13A894CF68ADDD849A2F63FE03F2DCDD
A>> T=1 (4+0075) 84D80181 4B
↪  0180104B5D0DA613A894CF68ADDD849A2F63FE03F2DCDD80104B5D0DA
613A894CF68ADDD849A2F63FE03F2DCDD80104B5D0DA613A894CF68AD
DD849A2F63FE03F2DCDDBE99A74027AB9365
```

```
A<< (0010+2) (25ms) 01F2DCDDF2DCDDF2DCDD 9000
Card locked with: 010B0371D78377B801F2D62AFC671D95
Write this down, DO NOT FORGET/LOSE IT!
```

Dodatkowo, wykorzystano możliwości JCIDE, które pozwala na sprawdzenie historii otrzymywanych poleceń oraz zwracanych odpowiedzi do zweryfikowania poprawności wysyłanych poleceń:

```
>> 00 A4 04 00 00 (APDU CASE2)
<< 6F 5C 84 08 A0 00 00 00 03 00 00 00 A5 50 73 4A 06 07 2A 86 48 86 FC 6B 01 60
↪  0C 06 0A 2A 86 48 86 FC 6B 02 02 01 01 63 09 06 07 2A 86 48 86 FC 6B 03 64 0B
↪  06 09 2A 86 48 86 FC 6B 04 02 15 65 0B 06 09 2B 85 10 86 48 64 02 01 03 66 0C
↪  06 0A 2B 06 01 04 01 2A 02 6E 01 02 9F 65 01 FF 90 00

>> 80 50 00 00 08 8A 7C 02 D6 AF F1 2B 5B 00 (APDU CASE4)
<< 00 00 00 00 00 00 00 00 00 00 01 02 00 00 3D 02 9C 31 C7 89 9C 6F 63 1B 14 7B
↪  3E 1A 90 00

>> 84 82 01 00 10 15 4A 72 DB D0 BC 5F 1E E1 11 AF 9A 8C 97 B7 47 (APDU CASE3)
<< 90 00

>> 84 CA 00 E0 08 D0 70 0E 7D 42 7F 32 78 00 (APDU CASE4)
<< E0 12 C0 04 01 01 80 10 C0 04 02 01 80 10 C0 04 03 01 80 10 90 00

>> 84 CA 00 E0 08 7D A7 E0 ED 3C 1D 52 A9 00 (APDU CASE4)
<< E0 12 C0 04 01 01 80 10 C0 04 02 01 80 10 C0 04 03 01 80 10 90 00

>> 84 D8 01 81 4B 01 80 10 4B 5D 0D A6 13 A8 94 CF 68 AD DD 84 9A 2F 63 FE 03 F2
↪  DC DD 80 10 4B 5D 0D A6 13 A8 94 CF 68 AD DD 84 9A 2F 63 FE 03 F2 DC DD 80 10
↪  4B 5D 0D A6 13 A8 94 CF 68 AD DD 84 9A 2F 63 FE 03 F2 DC DD BE 99 A7 40 27 AB
↪  93 65 (APDU CASE3)
<< 01 F2 DC DD F2 DC DD F2 DC DD 90 00
```

5.5.3 Elektroniczna Legitymacja Studencka

Jako drugie urządzenie została użyta Elektroniczna Legitymacja Studencka, co pozwoliło sprawdzić przebieg procesu zmiany klucza w przypadku użycia protokołu SCP01. Podobnie jak w przypadku programowego rozwiązania, została wywołana poleceniem z opcją –lock w celu ustawienia nowej wartości klucza:

```
.\gp.exe -d --lock 010B0371D78377B801F2D62AFC671D95 --reader "Generic EMV
↪  Smartcard Reader 0"
...
A>> T=1 (4+0000) 00A40400 00
A<< (0111+2) (29ms) 6F6D8407A0000001510000A562732F06072A864886FC6B01600C060A2
A864886FC6B020201016309060 72A864886FC6B03640B06092A864886
FC6B0401059F6E2A479051688231932200 72807747FBEA66750011428
1731143817311448173141000000000000000000009F6501FF 9000
...
[DEBUG] GPSession - Auto-detected ISD: A0000001510000
Warning: no keys given, using default test key 404142434445464748494A4B4C4D4E4F
...
[TRACE] GPSession - Generated host challenge: CFD315D2C72EE563
A>> T=1 (4+0008) 80500000 08 CFD315D2C72EE563 00
A<< (0028+2) (38ms) FF998886000047FBEA66010189223689C5B785DE9CC6BAA92FD6537F 9000
[DEBUG] GPSession - Host challenge: CFD315D2C72EE563
[DEBUG] GPSession - Card challenge: 89223689C5B785DE
[DEBUG] GPSession - Card reports SCP01 with key version 1 (0x01)
...
[INFO] GPSession - Session keys:
ENC=C46546CC4F18189B567646C0FFB66DD0
MAC=C46546CC4F18189B567646C0FFB66DD0
...
[DEBUG] GPSession - Verified card cryptogram: 9CC6BAA92FD6537F
[DEBUG] GPSession - Calculated host cryptogram: 174621526F3E2546
A>> T=1 (4+0016) 84820100 10 174621526F3E254691C1F0129EA82907
A<< (0000+2) (29ms) 9000
A>> T=1 (4+0008) 84CA00E0 08 A8A743FC83FCCD93 00
A<< (0020+2) (17ms) E012C00401018010C00402018010C00403018010 9000
...
[DEBUG] GPSession - PUT KEY version ENC=010B0371D78377B801F2D62AFC671D95 (KCV:
↪  F2DCDD) MAC=010B0371D78377B801F2D62AFC671D95 (KCV: F2DCDD)
↪  DEK=010B0371D78377B801F2D62AFC671D95 (KCV: F2DCDD) for null
```

```
A>> T=1 (4+0008) 84CA00E0 08 5B515BFB0A801AA8 00
A<< (0020+2) (17ms) E012C00401018010C00402018010C00403018010 9000
...
[DEBUG] PlaintextKeys - Encrypting ENC value 010B0371D78377B801F2D62AFC671D95
↪  with 404142434445464748494A4B4C4D4E4F
...
A>> T=1 (4+0075) 84D80181 4B
↪  01801001F2D62AFC671D9575C4C08ED4FCC69303F2DCDD801001F2D62AFC
671D9575C4C08ED4FCC69303F2DCDD801001F2D62AFC671D9575C4C08ED4
FCC69303F2DCDDCFC44EA03D7BCE0F
A<< (0010+2) (340ms) 01F2DCDDF2DCDDF2DCDD 9000
Card locked with: 010B0371D78377B801F2D62AFC671D95
Write this down, DO NOT FORGET/LOSE IT!
```

5.5.4 Wnioski

Testy z wykorzystaniem aplikacji GlobalPlatformPro pozwoliły potwierdzić, że zarówno karta elektroniczna wykorzystująca protokół SCP01 oraz emulowane środowisko wpierające obsługę protokołu SCP02 są odpowiednie do wykorzystania w dalszych częściach pracy. Narzędzie to pozwoliło również poznać przebieg wykonywanych obliczeń oraz wysyłanych poleceń pomiędzy urządzeniem a kartą w celu potwierdzenia schematu działania wymiany klucza opisanego w części teoretycznej pracy.

5.6 Tajne klucze

Zgodnie ze specyfikacją PKCS#11 [30], klasa obiektu CKO_SECRET_KEY pozwala na przechowywanie tajnych kluczy z możliwością zdefiniowania ich wartości w momencie tworzenia obiektu. Dzięki wykorzystaniu możliwości biblioteki Pkcs11interop możliwe było generowanie obiektów na karcie. Metoda CreateObject tworzy nowy tajny obiekt klucza 3DES przechowywany na tokenie na podstawie określonych atrybutów. Tak utworzony obiekt, może być wykorzystywany do wykonywania operacji szyfrowania oraz deszyfrowania danych. Nazwa klucza oraz jego wartość są definiowane przez użytkownika za pomocą atrybutów CKA_VALUE oraz CKA_LABEL:

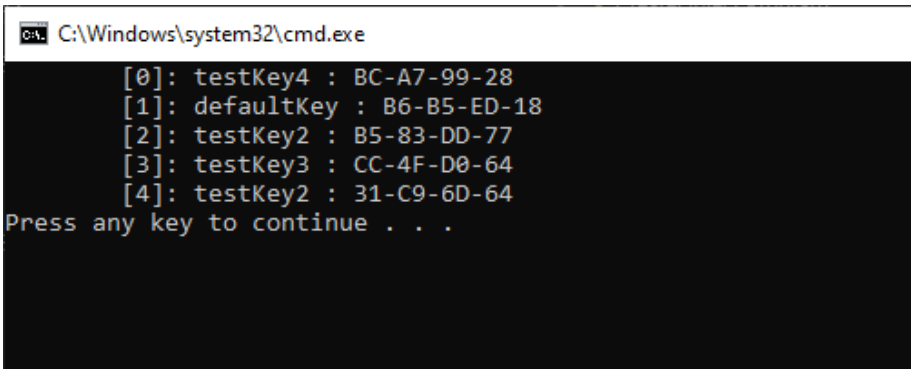
```
List<IObjectAttribute>OA=newList<IObjectAttribute>();
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_TOKEN,true));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_CLASS,CKO.CKO_SECRET_KEY));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_KEY_TYPE,CKK.CKK_DES3));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_SENSITIVE,false));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_EXTRACTABLE,true));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_ENCRYPT,true));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_DECRYPT,true));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_VALUE,keyValue));
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_LABEL,keyLabel));
byte[]ID=newbyte[4];
newRandom().NextBytes(ID);
OA.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_ID,ID));

session.CreateObject(OA);
```

Druga część programu odpowiadała za prezentowanie dostępnych tajnych kluczy znajdujących się na urządzeniu. Ponownie została wykorzystana biblioteka Pkcs11interop pozwalająca na wyszukiwanie kluczy znajdujących się na tokenie w oparciu o zdefiniowane atrybuty:

```
List<IObjectAttribute> SK = new List<IObjectAttribute>();
SK.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_CLASS,CKO.CKO_SECRET_KEY));
SK.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_KEY_TYPE,CKK.CKK_DES3));
SK.Add(session.Factories.ObjectAttributeFactory.Create(CKA.CKA_TOKEN,true));

List<CKA> attributes = new List<CKA>();
attributes.Add(CKA.CKA_LABEL);
attributes.Add(CKA.CKA_ID);
List<IObjectHandle> foundObjects = session.FindAllObjects(SK);
if (foundObjects.Count < 1)
{
  Console.WriteLine("Key(s) not found!");
  Environment.Exit(0);
}
else
{
  int keyList = 0;
  foreach (var objectKey in foundObjects)
  {
    var objectKeyAttributes = session.GetAttributeValue(objectKey, attributes);

    Console.WriteLine("\t[{0}]: {1} : {2}",
          keyList.ToString(),
          objectKeyAttributes[0].GetValueAsString(),
          BitConverter.ToString(objectKeyAttributes[1].GetValueAsByteArray()));

    keyList++;
  }
}
```

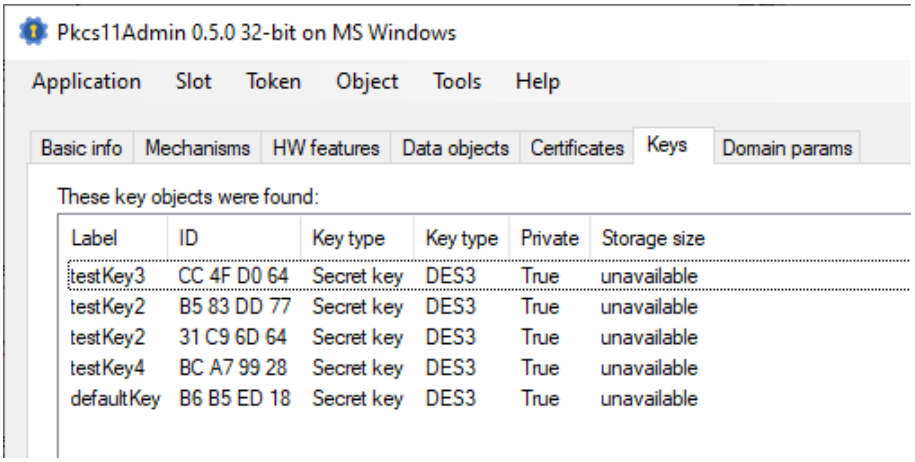
5.6.1 SoftHSM

Uruchomienie przygotowanego kodu, potwierdziło możliwości SoftHSM do przechowywania wielu tajnych obiektów. Na rysunku 5.8 przedstawiono zrzut ekranu z programu potwierdzający umieszczenie nowych obiektów na programowym module bezpieczeństwa.



Rysunek 5.8: Lista tajnych obiektów przechowywanych na tokenie

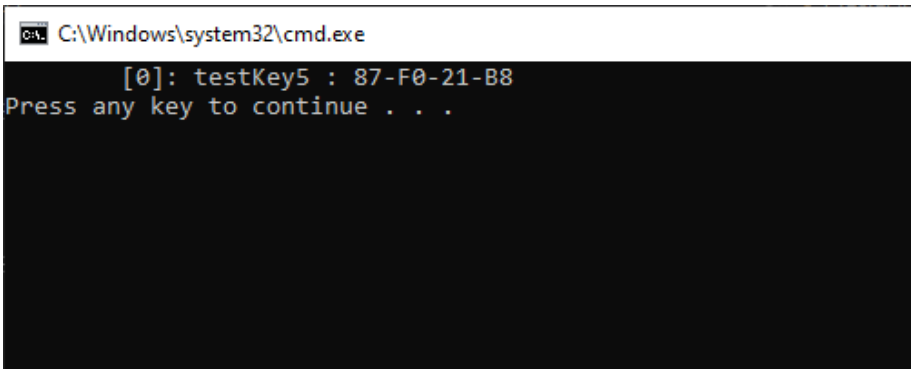
Do potwierdzenia poprawności tworzonych obiektów, użyto również aplikację Pkcs11Admin. Na rysunku 5.9 zaprezentowano zrzut ekranu pokazujący tajne klucze znajdujące się na urządzeniu.



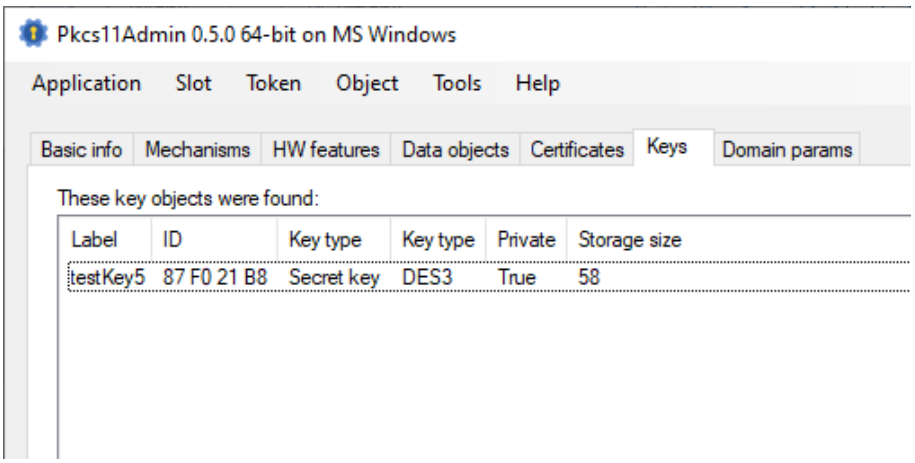
Rysunek 5.9: Lista tajnych obiektów przechowywanych na tokenie zaprezentowana w programie Pkcs11Admin

5.6.2 Oberthur Cosmo64 v5.4

W przypadku karty elektronicznej Oberthur Cosmo64 v5.4 testy potwierdziły mozliwość przechowywania tylko jednego tajnego obiektu. Próba utworzenia więcej niż jednego obiektu powodowała uszkodzenie znajdującego się na urządzeniu aktualnego obiektu. Ze względu na brak oficjalnych informacji od producenta karty, dotyczących możliwości przechowywania tajnych kluczy na karcie nie określono czy błąd powodowała wykorzystywana biblioteka PKCS#11, czy były to ograniczenia karty elektronicznej. Na rysunkach 5.10 oraz 5.11 zaprezentowano pomyślne utworzenie jednego tajnego klucza 3DES o zdefiniowanej przez użytkownika wartości.



Rysunek 5.10: Lista tajnych obiektów przechowywanych na tokenie



Rysunek 5.11: Lista tajnych obiektów przechowywanych na tokenie zaprezentowana w programie Pkcs11Admin

W przypadku próby utworzenia kolejnego obiektu, program zwracał błąd CKR_GENERAL_ERROR:

```
Net.Pkcs11Interop.Common.Pkcs11Exception:
Method C_CreateObject returned CKR_GENERAL_ERROR
at Net.Pkcs11Interop.HighLevelAPI41.Session.CreateObject(List`1 attributes)
```

5.6.3 Wnioski

W związku z ograniczeniami karty elektronicznej, w dalszej części pracy badawczej zostało wykorzystane programowe rozwiązanie SoftHSM. Pozwoliło to uprościć proces tworzenia rozwiązania. Możliwość utworzenia tylko jednego klucza na jednym urządzeniu wymagałaby posiadania przynajmniej dwóch czytników oraz dwóch kart elektronicznych podłączonych w tym samym czasie do komputera. Dodatkowo zwiększyłoby to trudność w programowaniu rozwiązania ze względu na konieczność przełączania pomiędzy kilkoma jednocześnie otwartymi sesjami utworzonymi przez oprogramowanie z kartami elektronicznymi.

5.7 Proponowane rozwiązanie

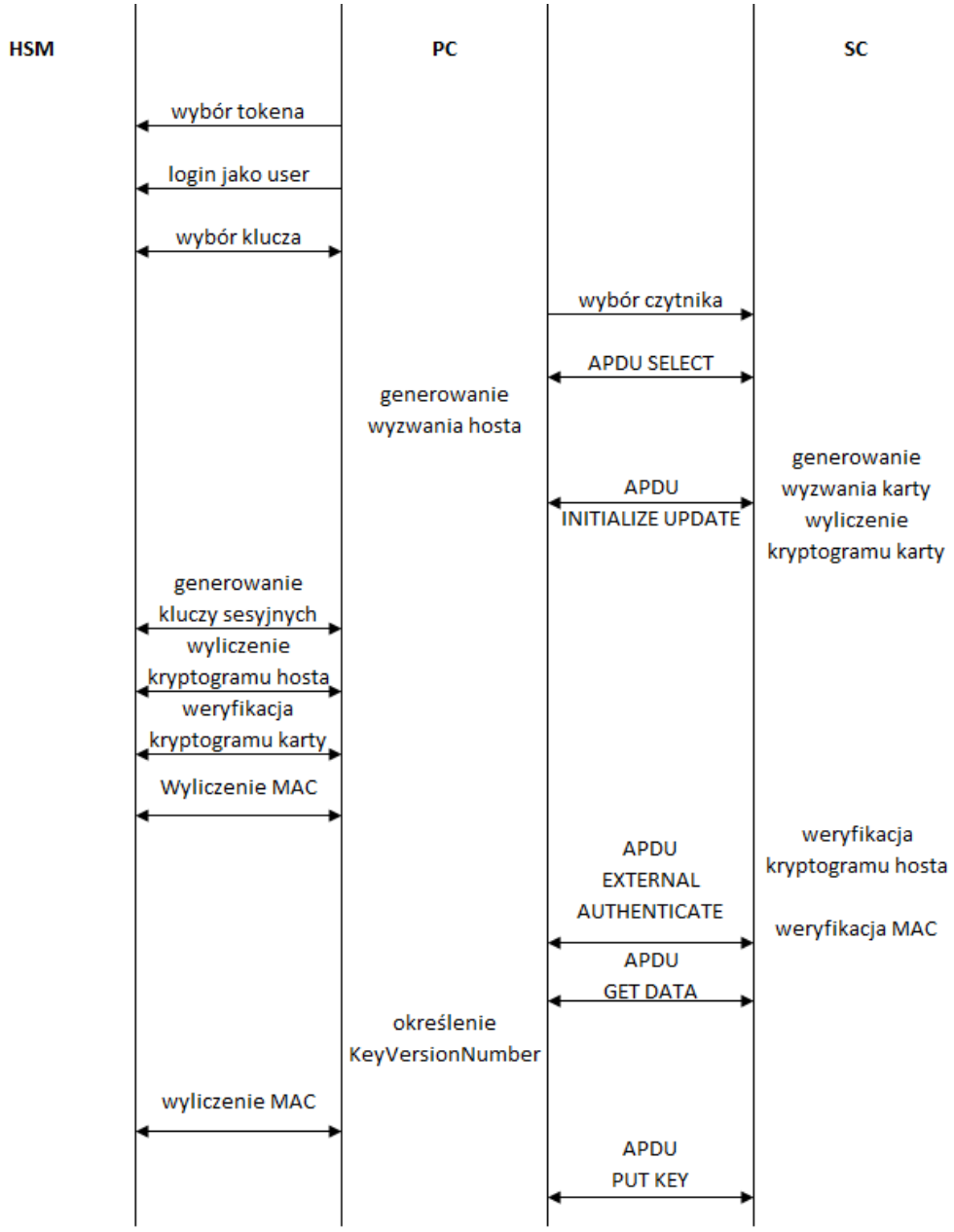
Ideą przygotowanego rozwiązania jest wykorzystanie karty elektronicznej jako urządzenia HSM do bezpiecznego przechowywania kluczy kryptograficznych. Klucze te są następnie używane w procesie wymiany kluczy dla domeny bezpieczeństwa na innej karcie bezpieczeństwa.

Do zarządzania kluczami znajdującymi się na karcie wykorzystywanej jako HSM wykorzystano możliwości biblioteki Pkcs11interop. Umożliwiła ona bezpieczny dostęp do obiektów znajdujących się na karcie oraz przeprowadzanie odpowiednich operacji kryptograficznych wymaganych w procesie wymiany klucza. W celu wywołania wymaganych poleceń APDU na karcie docelowej została użyta biblioteka pcsc-sharp. Pozwoliła ona na wysłanie odpowiednich poleceń zezwalających na uwierzytelnianie się karty oraz hosta, a następnie wymianę kluczy.

Przed uruchomieniem aplikacji niezbędne jest wskazanie biblioteki PKCS#11 dostarczonej przez producenta karty elektronicznej, która ma zostać wykorzystana jako urządzenie HSM. Zostaje ona podana jako jeden z parametrów podczas ładowania oraz inicjalizacji nowej instancji klasy IPkcs11Library. Należy również wskazać czytnik kart elektronicznych, w którym znajduje się karta. Z interfejsu ISlot, zostaje wywołana metoda OpenSession() w celu otwarcia sesji pomiędzy aplikacją a tokenem w danym slocie oraz metoda GetSlotInfo() do wyświetlenia właściwości tego slotu. W celu uzyskania dostępu do obiektów znajdujących się na karcie należy wprowadzić kod PIN użytkownika zdefiniowany przez administratora karty. Następuje wywołanie metody Login() dla wybranej sesji. Karty dostarczają silne zabezpieczenia w procesie uwierzytelniania, dzięki czemu można zagwarantować, że tylko uprawnione osoby uzyskają dostęp do obiektów na nich przechowywanych. Podanie poprawnego kodu PIN pozwala na uwierzytelnianie się do karty oraz wykonywanie operacji kryptograficznych. Zgodnie z zasadami bezpieczeństwa zdefiniowanymi na karcie elektronicznej, w przypadku kilkukrotnego niepoprawnego wprowadzenia kodu PIN, kod zostanie zablokowany, a użytkownik straci możliwość uwierzytelniania się. W takim przypadku będzie konieczne odblokowanie karty przez administratora karty, który dzięki kodowi PUK będzie mógł ponownie ustawić kod PIN. Kolejne nieudane próby wprowadzenie kodu PUK mogą doprowadzić do zablokowania karty, a tym samym do uniemożliwienia uzyskania dostępu do obiektów na niej przechowywanych. Stworzona aplikacja nie zawiera metod do zarządzenia kodami PIN oraz PUK na karcie elektronicznej. W celu zmiany kodu PIN należy skorzystać z oprogramowania dostarczonego przez producenta karty lub oprogramowania firm trzecich.

Kolejnym krokiem jest wybór czytnika kart, w którym znajduje się karta, na której ma zostać dokonana operacja zmiany klucza. Do obsługi karty docelowej zostaje wykorzystana biblioteka pcsc-sharp. Wzajemne uwierzytelnianie jest osiągane poprzez proces inicjowania Bezpiecznego Kanału. W tym celu należy wskazać klucz znajdujący się na HSM, który jest aktualnie używany przez kartę. Proces ten obejmuje tworzenie nowych wyzwań i kluczy sesji Bezpiecznego Kanału. Jeżeli którykolwiek z etapów procesu wzajemnego uwierzytelniania zakończy się niepowodzeniem, proces ten zostanie zakończony. Będzie wymagane ponownie wykonanie procesu, tzn. zostaną wygenerowane nowe wyzwania i klucze sesji. Należy również wybrać nowy klucz, który będzie

wykorzystywany przez domenę bezpieczeństwa po poprawnym wywołaniu polecenia PUT KEY na karcie docelowej. Na rysunku 5.12 można zobaczyć przepływ wymiany poleceń, niezbędnych do poprawnego wykonania polecenia PUT KEY z wykorzystaniem kluczy znajdujących się na urządzeniu typu HSM.



Rysunek 5.12: Schemat przebiegu procesu

Pierwszym krokiem jest wysłanie polecenia SELECT bez podanego AID w celu wyboru domeny bezpieczeństwa. Następnie następuje wygenerowanie wyzwania hosta. Do tego celu zostaje wykorzystana klasa RNGCryptoServiceProvider, która implementuje kryptograficzny generator liczb losowych (RNG), dzięki czemu można zapewnić odpowiednią losowość wygenerowanego wyzwania.

Wygenerowane wyzwanie hosta użyte zostaje w procesie inicjalizacji sesji Bezpiecznego Kanału. Wysyłane zostaje polecenie INITIALIZE UPDATE służące do przesyłania danych karty i Sesji Bezpiecznego Kanału pomiędzy kartą a hostem. W odpowiedzi uzyskujemy dane dywersyfikacyjne, numer wersji klucza oraz identyfikator Protokołu Bezpiecznego Kanału, wygenerowane przez kartę losowo wyzwanie oraz kryptogram karty.

Otrzymane dane pozwalają na określenie, który Protokół Bezpiecznego Kanału jest wspierany przez kartę docelową. W zależności od zwróconego rezultatu następuje wybór pomiędzy wersją SCP01 lub SCP02, które zostały zaimplementowane w przygotowanej aplikacji. Ze względu na brak dostępnego programowego rozwiązania pozwalającego sprawdzić możliwości protokołu SCP03, jego obsługa nie została dodana do tworzonego oprogramowania.

5.7.1 SCP01

Do sprawdzenia poprawności wykonywanych obliczeń w przypadku protokołu SCP01 została wykorzystana Elektroniczna Legitymacji Studencka. Domena Bezpieczeństwa posiada trzy identyczne 16-bajtowe klucze statyczne. Poniżej zaprezentowano przebieg procesu:

1. Po wybraniu wykorzystywanych kluczy, następuje wywołanie polecenia SELECT, które prowadzi do wybrania domeny bezpieczeństwa. Zgodnie ze specyfikacją GlobalPlatform, zostaje wybrany domyślny AID dla domeny bezpieczeństwa wystawcy, A0000001510000.

   ```
   SELECT
   Send APDU command: 00A4040000
   SW1 SW2 = 90 00
   Challenge: 6F6D8407A0000001510000A562732F06072A864886FC6B0
   1600C060A2A864886FC6B02020101630906072A864886FC6B03640B060
   92A864886FC6B0401059F6E2A479051688231932200728077 47FBEA667
   500114281731143817311448173141000000000000000000000009F6501FF
   Host challenge: CE423953B0CC6D42
   ```

2. W następnej kolejności następuje wysłanie polecenia INITIALIZE UPDATE:

   ```
   INITIALIZE UPDATE
   Send APDU command: 8050000008CE423953B0CC6D4200
   SW1 SW2 = 90 00
   Challenge: FF998886000047FBEA6601017F1A61186B1E452BFAA236EC095F6436
   Card challenge: 7F1A61186B1E452B
   Card cryptogram: FAA236EC095F6436
   ```

3. Wyzwania hosta oraz karty zostają wykorzystane do utworzenia 16-bajtowej tablicy, która zawiera dane dywersyfikacyjne.

   ```
   Derivation data: 6B1E452BCE4239537F1A6118B0CC6D42
   ```

4. Na podstawie tych danych, następuje wygenerowanie kluczy sesyjnych S-ENC oraz S-MAC. Do obliczenia ich wartości, zostaje wykorzystana biblioteka Pkcs11interop, implementująca klasę Net.Pkcs11Interop.HighLevelAPI, w której zawarta jest metoda Encrypt(). Jako mechanizm zostaje wykorzystany DES w trybie ECB. Obiektem jest klucz znajdujący się na karcie, a jako dane są podawane dane dywersyfikacyjne, które mają zostać poddane szyfrowaniu.

   ```
   sessionKey_Encryption_Value: 3CDAE9300BA6B5928E2F03951AB586CF
   sessionKey_S_MAC_Value: 3CDAE9300BA6B5928E2F03951AB586CF
   ```

5. Obliczone klucze zostają użyte w celu przeprowadzenia weryfikacji kryptogramu karty do potwierdzenia, że został określony prawidłowy klucz, który jest aktualnie używany przez kartę docelową. Jeżeli weryfikacja się nie powiedzie, program zakończy swoje działanie. W przypadku prawidłowego rezultatu, zostanie wygenerowany kryptogram hosta.

   ```
   Verify card cryptogram - OK
   Host cryptogram: 43271B3EF80EA58B
   ```

6. Wywołane zostaje polecenie EXTERNAL AUTHENTICATE w celu uwierzytelniania hosta i określenia poziomu bezpieczeństwa wymaganego dla wszystkich kolejnych poleceń.

   ```
   EXTERNAL AUTHENTICATE
   Send APDU: 848201001043271B3EF80EA58B828B6AD5FFC7D627
   SW1 SW2 = 90 00
   No data
   ```

7. W kolejnej części działania programu, zostaje wysłane polecenie GET DATA w celu określenia aktualnego numeru wersji klucza. Ze względu na konieczność użycia MAC w wysyłanych poleceniach, została napisana metoda, która wykorzystuje HSM oraz klucz sesji do jego obliczenia w celu zapewnienia odpowiedniego poziomu bezpieczeństwa dla wysyłanych poleceń APDU.

   ```
   GET DATA
   Send APDU command: 84CA00E0087EEFFD96F8A8277700
   SW1 SW2 = 90 00
   Challenge: E012C00401018010C00402018010C00403018010
   ```

```
GET DATA
Send APDU command: 84CA00E0086FDD085C41FBE7A800
SW1 SW2 = 90 00
Challenge: E012C00401018010C00402018010C00403018010
KeyVersionNumber: 1
```

8. Ostatnim krokiem jest przygotowanie danych, które maja zostać przekazane w poleceniu PUT KEY. W tym celu wartość nowego klucza zostaje poddana szyfrowaniu za pomocą klucza znajdującego się na HSM, a następnie dane zostają złożone w całość i wysłane do karty docelowej za pomocą polecenia APDU. Powodzenie wykonania polecenia jest sygnalizowane za pomocą odpowiedzi 90 00 zwróconej do urządzenia hosta.

```
PUTKEY
Send APDU command:
84D801814B01
801001F2D62AFC671D9575C4C08ED4FCC69303F2DCDD
801001F2D62AFC671D9575C4C08ED4FCC69303F2DCDD
801001F2D62AFC671D9575C4C08ED4FCC69303F2DCDD
5C03DA9E96828A4000
SW1 SW2 = 90 00
Challenge: 01F2DCDDF2DCDDF2DCDD
```

5.7.2 SCP02

W przypadku wykorzystania protokołu SCP02, do potwierdzenia poprawności przeprowadzonych obliczeń zostało wykorzystane oprogramowanie JCIDE oraz wirtualny czytnik kart utworzony w czasie instalacji JCKit. Poniżej zademonstrowano działanie aplikacji:

1. Podobnie jak w przypadku SCP01, na początku procesu wymiany kluczy, zostają wysłane polecenia SELECT oraz INITIALIZE UPDATE do karty docelowej:

```
SELECT
Send APDU command: 00A4040000
SW1 SW2 = 90 00
Challenge:
↪  6F5C8408A0000000030000000A550734A06072A864886FC6B01600C060A2A864886FC6
B020201016309060 72A864886FC6B03640B06092A864886FC6B040215650B06092B8510864864020
103660C060A2B060104012A026E01029F6501FF
Host challenge: 514EB0BD420CA4DC

INITIALIZE UPDATE
Send APDU command: 8050000008514EB0BD420CA4DC00
SW1 SW2 = 90 00
Challenge: 00000000000000000000010200003D029C31C789D2CF00BF8F2B9CE5
Card challenge: 00003D029C31C789
Card cryptogram: D2CF00BF8F2B9CE5
```

2. Następuje obliczenie kluczy sesyjnych wymaganych w dalszych etapach. Zastosowanie odpowiednich metod, pozwala na wykorzystanie klucza głównego znajdującego się na HSM do zaszyfrowania danych dywersyfikacyjnych z wykorzystaniem DES3 w trybie CBC. Dane te są tworzone na podstawie odpowiedzi polecenia INITIALIZE UPDATE oraz reguł opisanych w dokumentacji Global Platform.

```
C-MAC Session Key : D1C28C601652A4770D67AD82D2D2E1C4
C-MAC Session Key : D1C28C601652A4770D67AD82D2D2E1C4
R-MAC Session Key : FFAEC7EC7FAD69F9FBFF093BF2F79C45
Encryption Session Key : 010B0371D78377B801F2D62AFC671D95
Data Encryption Session Key : E11987EE331B417A5D67D760692F89D4
```

3. Odpowiedni klucz sesyjny wykorzystany zostaje w celu wygenerowania kryptogramu hosta, który będzie wymagany przy wywołaniu polecenia EXTERNAL AUTHENTICATE.

```
Host cryptogram: 40F57BE3E8BE5DEAE8C2762C3C0E1230D1F9141FA25FFA3F
```

4. Podobnie jak w SCP01, w celu potwierdzenia, że został wykorzystany prawidłowy, aktualny klucz główny, zostaje wykonana operacja sprawdzenia wartości kryptogramu. W przypadku niepowodzenia, program kończy swoje działanie. Prawidłowa weryfikacja, pozwala na kontynuowanie działania programu.

```
Host cryptogram: 40F57BE3E8BE5DEAE8C2762C3C0E1230D1F9141FA25FFA3F
Verify card cryptogram: D2CF00BF8F2B9CE5 - OK
```

5. Zostają przygotowane dane, a następnie wykonane polecenie EXTERNAL AUTHENTICATE do uwierzytelniania hosta oraz określenia poziomu bezpieczeństwa wymaganego dla wszystkich kolejnych poleceń.

```
EXTERNAL AUTHENTICATE
Send APDU: 8482010010D1F9141FA25FFA3F495312A930A9C6AB
SW1 SW2 = 90 00
No data
```

6. Na podstawie odpowiedzi GET DATA, zostaje określony numer aktualnej wersji klucza. W celu obliczenia MAC, została napisana metoda pozwalająca na wykonanie szyfrowania Single DES Plus Final Triple DES, znanego również jako Retail MAC.

```
Send APDU command: 84CA00E008B7CD91405150B20800
SW1 SW2 = 90 00
Challenge: E012C00401018010C00402018010C00403018010


Send APDU command: 84CA00E00862F674F1C760AC0600
SW1 SW2 = 90 00
Challenge: E012C00401018010C00402018010C00403018010
```

7. Wygenerowany wcześniej klucz sesji zostaje wykorzystany do zaszyfrowania wartości nowego klucza, który ma zostać ustawiony na karcie docelowej. Zostają przygotowane dane, a następnie wywołane polecenie PUT KEY. Powodzenie wykonania jest określane za pomocą odpowiedzi 90 00 zwróconej do hosta.

```
Send APDU command:
    84D801814B01
    80104B5D0DA613A894CF68ADDD849A2F63FE03F2DCDD
    80104B5D0DA613A894CF68ADDD849A2F63FE03F2DCDD
    80104B5D0DA613A894CF68ADDD849A2F63FE03F2DCDD
    5B3032B5AEA56F0C00
SW1 SW2 = 90 00
Challenge: 01F2DCDDF2DCDDF2DCDD
```

5.8 Wykorzystane metody

Na potrzeby przeprowadzania obliczeń wymaganych w procesie wymiany kluczy, zostały utworzonego metody wykonujące odpowiednie operacje kryptograficzne.

5.8.1 calculcateDESSession

Klucze sesji DES są generowane za każdym razem, gdy inicjowany jest bezpieczny kanał w celu zapewnienia, że dla każdej bezpiecznej sesji komunikacyjnej używany jest inny zestaw kluczy. Te klucze sesji mogą być używane do kolejnych poleceń, jeśli wymagana jest bezpieczna wymiana komunikatów. W tym celu została stworzona metoda do ich obliczania. Jako parametry wejściowe przyjmuje ona aktualny klucz główny karty, odpowiedź uzyskaną po wywołaniu polecenia INITIALIZE UPDATE oraz stałą zdefiniowaną w dokumentacji GlobalPlatform. Odpowiedź jest wykorzystywana do przygotowania danych dywersyfikacyjnych, a stała określa typ tworzonego nowego klucza. Tryb DES używany do generowania tych kluczy to DES3 w trybie CBC.

Algorytm 1 calculcateDESSessionKeyValue(session,INITIALIZE_UPDATE,constant,masterKey)

$encryptionMechanism \leftarrow CKM\_DES3\_CBC, new\ byte[8]$
$sequenceCounter \leftarrow Array.Copy(INITIALIZE\_UPDATE, 12, sequenceCounter, 0, 2);$
$derivationData \leftarrow constant\ (2bytes) \mid sequenceCounter\ (2bytes) \mid '00'\ Padding\ (12bytes)$
$sessionKey \leftarrow Encrypt(encryptionMechanism, masterKey, derivationData)$
return $sessionKey$

5.8.2 Calculate MAC

C-MAC jest generowany przez hosta i jest stosowany w całym poleceniu APDU przekazywanym do karty. Do przeprowadzenia operacji kryptograficznych wykorzystywana jest karta elektroniczna pełniąca rolę HSM. Metoda obliczania MAC jest zależna od wykorzystywanej wersji Protokołu Bezpiecznego Kanału. Zostały przygotowane dwie wersje, obsługujące zarówno SCP01, jak i SCP02. Sposoby działania przedstawiono poniżej:

SCP01

Dla SCP01, wykorzystywana jest metoda Full Triple DES z użyciem klucza sesji MAC i ICV dająca w rezultacie 8-bajtowy rezultat, którym jest C-MAC.

Algorytm 2 CalculcateMAC_SCP01(session,SMAC,data,iv)

$encryptionMechanism \leftarrow CKM\_DES3\_CBC, iv$
$MAC \leftarrow Encrypt(encryptionMechanism, SMAC, data)$
$result \leftarrow Array.Copy(MAC, MAC.Length - 8, result, 0, 8);$
return $result$

SCP02

W przypadku SCP02, generowanie i weryfikacja C-MAC wykorzystuje klucz sesji C-MAC, ICV oraz metodę Single DES Plus Final Triple DES. Jest również znana jako Retail MAC i jest zdefiniowana w ISO 9797-1 jako MAC Algorithm 3.

Algorytm 3 CalculcateMAC(session,CMAC,data,iv)

$encryptionMechanism \leftarrow CKM\_DES\_CBC, iv$
$key1Value \leftarrow Array.Copy(CMAC, 0, key1Value, 0, 8)$
$key2Value \leftarrow Array.Copy(CMAC, 8, key2Value, 0, 8)$
$key1 \leftarrow createObject(session, key1Value, "key1", DES, false)$
$key1 \leftarrow createObject(session, key2Value, "key2", DES, false)$
$encryptionMechanism \leftarrow DES\ CBC, iv$
$step1 \leftarrow Encrypt(encryptionMechanism, key1, data)$
$step2 \leftarrow Array.Copy(step1, step1.Length - 8, step2, 0, 8);$
$step3 \leftarrow Decrypt(encryptionMechanism, key2, step2)$
$DestroyObject(key1)$
$DestroyObject(key2)$
$result \leftarrow Encrypt(encryptionMechanism, key1, step3)$
return *result*

Rozdział 6

Wnioski

W niniejszej pracy magisterskiej zaprezentowano przykład zastosowania kart elektronicznych w procesie bezpiecznego przechowywania kluczy kryptograficznych oraz wymiany kluczy w środowisku kart elektronicznych. Praca zawiera teoretyczne i praktyczne sposoby implementacji operacji kryptograficznych oraz mechanizmy zarządzania cyklem życia kluczy kryptograficznych. W ramach części praktycznej potwierdzono możliwość wykorzystania rozwiązania programowego SoftHSM emulującego pracę sprzętowych modułów bezpieczeństwa do wykonania wymaganych operacji kryptograficznych. W przypadku karty elektronicznej Oberthur Cosmo64 v5.4 stwierdzono, że urządzenie pozwala na przechowywanie jedynie jednego obiektu przechowującego wartość klucza 3DES wykorzystywanego w procesie wymiany kluczy.

Podczas przeglądu dostępnych rozwiązań, nie znaleziono rozwiązania spełniającego wymagania Międzyuczelnianego Centrum Personalizacji Legitymacji Studenckiej. Utworzono aplikację, która potwierdziła możliwość użycia kluczy kryptograficznych przechowywanych na programowym rozwiązaniu SoftHSM. Zaprezentowano poprawne działanie algorytmów kryptograficznych biorących udział w procesie ustanawiania bezpiecznego kanału oraz w wymianie kluczy kryptograficznych na karcie. W ramach pracy zdefiniowano również metody niezbędne do przeprowadzenia operacji kryptograficznych. Proponowane rozwiązanie korzysta z funkcji przechowywania oraz zarządzania kluczami kryptograficznymi oferowanymi przed dostawców kart kryptograficznych. Można je zintegrować z programowymi urządzeniami emulującymi pracę modułu kryptograficznego. Ze względu na rosnące ryzyko z punktu widzenia bezpieczeństwa będzie wzrastać przyszłe zapotrzebowanie na bezpieczne sposoby przechowywania kluczy kryptograficznych.

Proponowane rozwiązanie działa poprawnie w środowisku testowym przy wykorzystaniu rozwiązań programowych. W celu implementacji aplikacji w środowisku produkcyjnym dla fizycznych egzemplarzy kart elektronicznych, konieczny byłoby dalszy rozwój oprogramowania. Wymagana byłaby również współpraca z producentami innych kart elektronicznych w celu potwierdzenia możliwości tworzenia oraz przechowywania określonej liczby obiektów przy wykorzystaniu dostarczonych bibliotek PKCS#11. Przygotowane oprogramowanie daje podstawy do dalszego rozwoju w celu przyszłego zastosowania karty elektronicznej jako sprzętowy modułu bezpieczeństwa w Międzyuczelnianym Centrum Personalizacji Legitymacji Studenckiej.

Literatura

[1] https://aventra.fi/webshop/index.php?route=product/productproduct_id=53
[dostęp: 26.03.2019].

[2] https://docs.microsoft.com/en-us/windows/desktop/seccng/cng-features
[dostęp: 11.03.2019].

[3] https://docs.microsoft.com/en-us/windows/desktop/seccng/cng-portal
[dostęp: 11.03.2019].

[4] https://docs.microsoft.com/en-us/windows/desktop/SecGloss/c-gly
[dostęp: 09.03.2019].

[5] https://docs.microsoft.com/pl-pl/windows/desktop/seccrypto/cryptographic-service-providers
[dostęp: 09.03.2019].

[6] https://download.oracle.com/otndocs/jcp/java_card_kit-2.2.1-fr-oth-JSpec
[dostęp: 23.05.2019].

[7] https://github.com/danm-de/pcsc-sharp
[dostęp: 23.05.2019].

[8] https://github.com/martinpaljak/GlobalPlatformPro
[dostęp: 23.05.2019].

[9] https://github.com/opendnssec/SoftHSMv2
[dostęp: 26.03.2019].

[10] https://github.com/OpenSC/OpenSC/wiki/Card-personalization
[dostęp: 28.03.2019].

[11] https://github.com/opensc/opensc/wiki
[dostęp: 28.03.2019].

[12] https://globalplatform.org/specs-library
[11.03.2019].

[13] https://globalplatform.org/specs-library/card-specification-v2-3-1
[dostęp: 11.03.2019].

[14] https://go.ncipher.com/rs/104-QOX-775/images/nCipher_nShield_Family_Brochure.pdf
[dostęp: 25.02.2019].

[15] http://shop.pivkey.com/c980-enterprise-dual-pki-smart-card
[dostęp: 22.03.2019].

[16] https://pkcs11admin.net
[dostęp: 23.05.2019].

[17] https://pkcs11interop.net
[dostęp: 23.05.2019].

[18] https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms/fips-common-criteria-validations
[dostęp: 25.02.2019].

[19] https://safenet.gemalto.com/data-encryption/hardware-security-modules-hsms
[dostęp: 25.02.2019].

[20] https://support.microsoft.com/en-us/help/909520/description-of-the-software-update-for-base-smart-card-cryptographic-s
[dostęp: 11.03.2019].

[21] https://www.acs.com.hk/en/products/308/acos5-64-v3.00-cryptographic-card-contact
[dostęp: 23.03.2019].

[22] https://www.cardomatic.de/epages/64510967.sf/en_GB/?ObjectPath=/Shops/64510967/Products/SmartCard-HSM-4K-Token
[dostęp: 14.03.2019].

[23] https://www.cardomatic.de/epages/64510967.sf/en_GB/=/Shops/64510967/Products/SmartCard-HSM-4K-Dual-IF
[dostęp: 11.03.2019].

[24] https://www.commoncriteriaportal.org
[dostęp: 27.02.2019].

[25] https://www.floss-shop.de/en/security-privacy/smartcards/13/openpgp-smart-card-v3.3
[dostęp: 20.03.2019].

[26] https://www.gemalto.com/companyinfo/smart-cards-basics
[dostęp: 11.05.2019].

[27] https://www.insight.com/en_US/shop/category/networking/store.html
[dostęp: 01.12.2019].

[28] https://www.iso.org/ics/35.240.15/x
[dostęp: 23.05.2019].

[29] https://www.javacardos.com/tools
[dostęp: 16.06.2019].

[30] https://www.oasis-open.org/standardspkcs11-base-v2.40
[dostęp: 09.03.2019].

[31] https://www.oberthur.com
[dostęp: 28.03.2019].

[32] https://www.pcisecuritystandards.org/pci_security/standards_overview
[dostęp: 25.02.2019].

[33] https://www.smartcard-hsm.com
[dostęp: 11.03.2019].

[34] https://www.thalesesecurity.com/faq/key-secrets-management/what-fips-140-2
[dostęp: 25.02.2019].

[35] https://www.yubico.com/product/yubihsm-2
[dostęp: 14.03.2019].

[36] http://www.cs.pl/produkty/elektroniczna-legitymacja-studencka
[dostęp: 11.03.2019].

[37] http://www.smartcardbasics.com/smart-card-security_3.html
[dostęp: 13.05.2019].

[38] Rozporządzenie Ministra Nauki i Szkolnictwa Wyższego z dnia 27 września 2018 r. w sprawie studiów.

[39] Shttps://www.smartcard-hsm.com/features.htmlusbstick
[dostęp: 11.03.2019].

[40] ttps://www.smartcard-hsm.com/features.html
[dostęp: 11.03.2019].

[41] The innovative smart card interface GemPC Twin. Technical report, 2003.

[42] Stanley R. Snouffer Annabelle Lee, Miles E. Smid. Security requirements for cryptographic modules. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 5 2001.