02. Nahrátí appletu na kartu, úkol na PIN

Budete potřebovat:

- Java SE (+NetBeans) – už máme z minula Stáhněte: archiv java_card_2021.zip (../files/java_card_2021.zip) s nástroji pro překlad, konverzi, nahrátí na kartu, testování
  - Rozbalte do X:\, vznikne tím adresář `x:\java_card` , na který se odkazují skripty níže. Balík obsahuje:
  - Java Card SDK 2.2.2
  - GPShell
  - JSmartCardExplorer
- Stáhněte archiv card_scripts_2021.zip (../files/card_scripts_2021.zip) se skripty pro překlad atd. v každém konkrétním projektu. Tento skript rozbalíte do kořene adresářové struktury projektu Netbeans tak, aby vytvořený adresář `card` byl na stejné úrovni jako existující `src` .

Úkoly

- Přeložte a nahrajte applet z minula na kartu, otestujte jeho funkčnost na kartě.
- Vytvořte nový applet, který bude
  - podporovat instrukce z minulého cvičení, ale některé z nich budou vyžadovat ověření PINem.
  - nastavovat PIN při instalaci appletu (parametry instalace)
  - ověřovat PIN pomocí VERIFY (INS=0x20), všechny instrukce kromě vrácení jména musejí zkontrolovat, zda je PIN ověřen
    - *návratový kód VERIFY bude buď 0x9000 (úspěch), nebo 0x6300 (chyba)*
  - pokud je počet neúspěšných pokusů 3 a víc, applet odmítne pracovat – při SELECT vrátí `SW_VERIFICATION_FAILED = 0x6300` , nebo alternativně se vůbec nevybere (viz metoda `select()` třídy `Applet`)
  - pokud není ještě PIN ověřen, instrukce, které jsou tím podmíněny, vrátí SW_PIN_VERIFICATION_REQUIRED = 0x6301
- Přeložte a nahrajte nový applet na kartu, otestujte jeho funkčnost na kartě.

Doporučení: Vytvořte si pro každou instrukci zvlášť metodu/funkci, která ji zpracuje. V metodě `process` pak bude základní ošetření hlavičky APDU, dekódování instrukcí a `switch` pro volání ostatních metod.

Práce s objektem OwnerPIN

API viz dokumentace (https://docs.oracle.com/javacard/3.0.5/api/javacard/framework/OwnerPIN.html) . Budeme potřebovat

- `OwnerPIN pin;`

    - `... pin = new OwnerPIN((byte) max_tries, (byte) max_len);` – deklarujte jako instanční proměnnou appletu, vytvořte při instalaci appletu
- `pin.update(bArray, bOffset, bLength);` – nastaví hodnotu PINu, nastavte při instalaci podle instalačních parametrů
- `boolean correct = pin.check(pole, (short) počátek, (byte) délka);` – kontrola PINu, nastaví interní flag
- `pin.isValidated()` - vrátí boolean, zda je v této session už PIN zkontrolován
- `pin.getTriesRemaining();` – vrací počet zbývajících pokusů. Pokud je 0, applet by měl být zablokován.
- Příklad z tutoriálu o psaní appletu (https://www.oracle.com/technetwork/java/embedded/javacard/documentation/intro-139322.html) je užitečný, až na získání instalačních parametrů (počáteční hodnoty PINu), které je popsáno níže:

Získání instalačních parametrů

Dokumentace API třídy Applet (https://docs.oracle.com/javacard/3.0.5/api/javacard/framework/Applet.html) obsahuje užitečný příklad, viz níže.

Naší funkci install přijdou parametry ve formě pole, indexu na začátek, a délky: `public static void install(byte[] bArray, short bOffset, byte bLength)`

Z těchto parametrů nás zajímá 3. položka (applet data), kde bude náš PIN, který jsme zadali při instalaci appletu. Položky jsou ukládány od bOffset ve formátu LV (length, value).

Formát instalačních dat (příklad, PIN=31323334, HEX):

… before bOffset	iLen	instance AID	cLen	control info	aLen	applet data
bOffset points →	07	01020304050607	02	1122	04	**31323334**

Všechny délky mohou být nulové.

Příklad kódu viz dokumentace API, třída Applet (https://docs.oracle.com/javacard/3.0.5/api/javacard/framework/Applet.html). Pozor, kód je nutno přizpůsobit naší situaci. **Doporučení**: Vytvořte konstruktor appletu s parametry stejnými, jako má metoda install, a strukturu dekódujte v konstruktoru.

Nastavení instalačních parametrů se provádí jinak pro simulátor a jinak potom na kartě.

Simulátor: na projektu pravé tlačítko, z kontextového menu Properties – Applets – Customize Instances – Deployment Parameters

Na (fyzické) kartě: v GlobalPlatform Pro (soubor install.cmd, viz níže) doplňte volbu `--params <hexstring>` (v GPShell je pro příkaz install volba `-instParam <hexstring>`)

Nahrávání a spouštění na kartě

Překlad a konverze

Pro nahrání appletu na kartu musíme mít zdrojový kód přeložený s příslušnou verzí JC SDK. Pro naše karty budeme používat převážně verzi 2.2.2. Buď máme příslušnou verzi integrovanou do NetBeans jako platformu (nemáme, tam je novější), nebo použijeme skript, například:

nastavení prostředí pro JCK 2.2.2, skript `e222.cmd`. Ověřte/změňte cesty k JAVA_HOME (JDK), JC_HOME (JCK), cestu k GPShell (kam jste rozbalili)

```
set JAVA_HOME=c:\PROGRA~1\Java\jdk1.8.0_181
set JC_HOME=x:\java_card\java_card_kit-2_2_2
path=%JAVA_HOME%\bin;%JC_HOME%\bin;x:\java_card;x:\java_card\GPShell-1.4.4;%path%
```

překlad a konverze, skript `c222.cmd`. Změňte názvy package a appletu, při konverzi se navíc volí nové AID appletu a package. (Zde **jiné**, než pro simulátor, ale můžete ho nechat).

```
set PACKAGE=classicapplet2
set APPLET=ClassicApplet2
javac -g -target 1.1 -source 1.2 -cp %JC_HOME%\lib\api.jar -d . ../src/%PACKAGE%/*.java
converter -exportpath %JC_HOME%\api_export_files -applet 0x01:0x02:0x03:0x04:0x5:0x6:0x8:0x9 %PACKAGE%.%APP
```

Konverzí vznikne v adresáři s přeloženým package (zde classicapplet2) podadresář javacard, kde nalezneme zkonvertovaný classicapplet2.cap. Ten se dá nahrát do karty.

Pozn.: Pokud vám converter hlás í stovky podivných chyb (nemůže nalézt proměnnou nebo metodu ve třídě), zkontrolujte, jestli v adresáři package není navíc nějaký .class soubor, který tam být nemá - converter totiž prohledává všechny .class soubory.

Tedy v krátkosti:

- Najděte na disku cestu ke svému projektu, kam jste rozbalili archiv 'card_scripts… .zip' výše. Editujte skript `c222.cmd` podle jmen v projektu.
- Otevřete příkazovou řádku `cmd` v adreáři `card`.
- Spusťte `e222.cmd`. Teď máte nastaveny proměnné prostředí. Platí jen v tomto okně.
- Spusťte `c222.cmd` pro překlad a konverzi appletu.

Nahrání na kartu

Pro nahrání appletu použijeme nástroj GlobalPlatform Pro (`gp.exe`), případně GPShell, které je obsaženy v archivu výše. V každém případě je potřeba respektovat předepsanou autentizační proceduru a znát klíč(e). Nepokoušejte se opakovaně

o autentizaci, která dopadla neúspěchem, protože riskujete zablokování karty vyčerpáním čítače neúspěšných pokusů. Pokud dojde k chybě při autentizaci, použijte způsob a klíč, který bude zaručeně fungovat, úspěšnou autentizací se čítač vymaže.

Použití skriptu pro GlobalPlatform Pro (doporučeno) K dispozici jsou skripty `install.cmd`, `list.cmd` a `delete.cmd`, které volají gp.exe s příslušnými parametry.

Podívejte se (a případně upravte) na obsah těchto skriptů. **Obvykle musíte editovat** `install.cmd`. Příkazem list vypište obsah karty (nahrané balíčky a applety), a porovnejte tento výpis před/po volání install a delete.

Testování appletu na kartě

Můžeme použít skript pro GPShell, příkaz `send_apdu`, viz skript `test.txt`. Spouštíme pomocí `test.cmd`.

Můžeme také použít GUI aplikaci např. JSmartCardExplorer. Další alternativou je napsat si program v Javě (pro PC) nebo v Pythonu, který bude s kartou komunikovat pod naší přímou kontrolou. Od Javy 1.6 je součástí API `javax.smartcardio`.