03. Šifrování, MAC kód

Vytvořte applet, který bude používat symetrickou šifru AES pro šifrování dat a zajištění integrity/autenticity dat. Váš applet bude šifrovat data, která jsou odeslána v příkazových APDU a vytvoří ověřovací kód zprávy (MAC). Také bude dešifrovat dříve zašifrovaná data a ověří MAC. Vaším úkolem je implementovat následující instrukce:

- INS = 0x42, která přijme data, která mají být šifrována, a vrátí zašifrovaná data společně s MAC. Maximální délka vstupních dat je 64 bajtů.
    - APDU odpověď bude obsahovat šifrový text následovaný MAC (to znamená x*16 bytů šifrovaných dat + 16 bytů MAC)
    - Pokud dojde k chybě, nevracejte žádná data, pouze ohlašte chybu v SW
- INS = 0x44, která přijme data, která mají být ověřena a dešifrována. Tato instrukce vrací dešifrovaná data. Maximální délka dešifrovaných vstupních dat je 80 bytů (64 bytů šifrovaných dat + 16 bytů MAC).
    - Pokud dešifrování nebo ověření MAC selže, nevracejte žádná data, pouze ohlašte chybu v SW

Šifra, která má být použita pro **šifrování/dešifrování, je AES-128 v režimu CBC**. Pro **MAC** použijeme **AES-128 také v režimu CBC**. Klíče pro šifrování i MAC mohou být pevně zakódovány, ale měly by být různé. IV lze také napevno zakódovat (nebo ponechat výchozí hodnotu - všechny nulové bajty). Šifra a MAC budou použity ve schématu **Encrypt then MAC** - tj. **Nejprve zašifrujeme data** a **poté vypočítáme MAC ze zašifrovaných dat**.

AES je bloková šifra, takže pokud ji používáme k šifrování dat, jejich délka musí být násobkem délky bloku (v tomto případě 16 bytů). Aby bylo možné zpracovávat data libovolné délky, je obvykle ke vstupním datům přidána výplň (padding). Abychom náš úkol zjednodušili, **nepoužíváme padding**, ale budeme očekávat, že data v příchozích APDU budou násobkem délky bloku. Pokud tomu tak není, odpovíme chybou.

Ještě jednou zdůrazňujeme, že nepoužití paddingu a IV je potenciální bezpečnostní chyba – v reálné aplikaci by bylo potřeba být mnohem důkladnější. Například bychom mohli generovat náhodný IV a realizovat padding typu PKCS#7 nebo Ciphertext Stealing. V této úloze to ale nepožadujeme.

Applet bude otestován na fyzické kartě a jeho testováním musíte prokázat jeho správnou funkčnost. Zde je seznam očekávaných odpovědí (SW) na příkaz APDU:

- Úspěch - ISO7816.SW_NO_ERROR (0x9000)
- Nesprávná CLA - ISO7816.SW_CLA_NOT_SUPPORTED (0x6E00)
- Nesprávná INS - ISO7816.SW_INS_NOT_SUPPORTED (0x6D00)
- Tělo příliš dlouhé - ISO7816.SW_WRONG_LENGTH (0x6700)
- Nesprávná délka dat, která mají být zpracována AES (nemáme padding) - ISO7816.SW_WRONG_LENGTH + CryptoException.getReason()
- Ověření MAC selže - ISO7816.SW_WRONG_DATA (0x6A80)

Použití šifry a MAC

Budeme pracovat s následujícími třídami:

- `AESKey`, který zapouzdřuje klíčový objekt, který bude vyžadován šifrou AES
  - `KeyBuilder` vytvoří klíč, např. `key_enc = (AESKey) KeyBuilder.buildKey (KeyBuilder.TYPE_xxx, KeyBuilder.LENGTH_xxx_xxx, false);`
  - metoda `setKey` nastaví klíč na definovanou hodnotu
- `Cipher` zastřešuje šifry
  - metoda `getInstance` vytvoří instanci šifrovacího objektu, např. `aes_enc = Cipher.getInstance(Cipher.ALG_AES_xxxxx, false);`
  - metoda `init` inicializuje šifru s objektem klíče, lze také použít k nastavení IV (nebo jiných specifických parametrů šifry)
  - metody `update` a `doFinal` šifrují/dešifrují vstupní data. Pro naše použití stačí metoda `doFinal`, protože nepotřebujeme zpracovávat data postupně v průběhu času. (Na rozdíl od např. OpenSSL, `doFinal` má i vstupní data.) `doFinal` šifruje/dešifruje vstupní data a vrací zpracovaná data. Metoda také vrací délku dat po zpracování.
    - Dávejte pozor na správné používání těchto metod. Pokud jsou použity nesprávně, vyvolá výjimku, kterou byste měli chytit (vložte tedy tyto metody do bloku try/catch). Např. pokud data nejsou zarovnána do bloku (nejsou vyplněna), `doFinal` vyvolá výjimku CryptoException.ILLEGAL_USE.
- `Signature` zastřešuje podpisové a MAC algoritmy (včetně HMAC, RSA, ECDSA apod.)
  - metoda `getInstance` vytvoří instanci podpisového objektu, musíte specifikovat algoritmus odpovídajícím způsobem našemu úkolu
  - `init` inicializuje instanci podpisu s klíčovým objektem, lze také použít k nastavení IV (nebo jiných parametrů specifických pro podpis/MAC)
  - `sign` vytvoří podpis/MAC ze vstupních dat. Můžeme také použít `update` pro delší zpracování dat v případě potřeby, ale v našem případě to není nutné. Tato metoda vrací délku podpisu/MAC a také samotný podpis/MAC.
    - Např. `short len_mac = aes_mac.sign(encrypted_data, (short) 0, len_enc, encrypted_data, len_enc);` - MAC vypočítáme ze šifrovaných dat a připojíme je do stejného pole hned za nimi.
  - `verify` ověřuje správnost podpisu/MAC. Vrací `true`, pokud je podpis/MAC ověřen, `false` jinak.

Viz [dokumentace (https://www.win.tue.nl/pinpasjc/docs/apis/jc222/)](https://www.win.tue.nl/pinpasjc/docs/apis/jc222/) pro další popis výše uvedených tříd a jejich metod (viz balíčky **javacard.security** a **javacardx.crypto**).