Zadání druhého projektu KRY 2025
Porovnání výměny klíčů: Diffie-Hellman vs. ECDH

V rámci projektu se zaměříte na implementaci a analýzu klasického Diffie-Hellmanova algoritmu pro výměnu klíčů (DH) a jeho varianty založené na eliptických křivkách (ECDH). Z projektu je možné získat celkem 10 bodů.

Shrnutí projektu (TL;DR)

Hlavní cíle:

- **Seznámení se s operacemi na eliptických křivkách** potřebných pro ECDH (sčítání bodů, násobení bodu skalárem, generování klíčů).
- **Seznámení se s algoritmy pro výměnu klíčů**: DH a ECDH výměna klíčů a výpočet sdíleného klíče.
- **Porovnání DH a ECDH**: analýza rychlosti algoritmů a bezpečnosti, velikosti klíčů apod.

Požadavky:

- **Implementace potřebných operací na eliptických křivkách** do připravené šablony v jazyku Python v3.8. Z této části lze získat až 3 body.
- **Implementace klient-server komunikace a výměny klíčů pomocí DH a ECDH**. Z této části lze získat až 4 body.
- **Analýza a porovnání DH a ECDH**. Odevzdání textové dokumentace a výsledků experimentů v rozsahu **maximálně** tři strany A4 ve formátu `pdf`. Z této části lze získat až 3 body.

Projekt odevzdávejte do e-learningu do **27. 4. 2025**. Odevzdáváte jeden soubor, a to archiv pojmenovaný vaším VUT číslem ve formátu `zip`, který obsahuje implementované řešení (všechny potřebné zdrojové Python soubory) a dokumentaci.

Konzultace k projektu poskytuje Ing. Vojtěch Staněk (`istanek@fit.vut.cz`).

1 Materiály k projektu

V e-learning modulu VUT IS máte k projektu k dispozici následující materiály:

- Soubor `zadani.pdf`, který právě čtete, obsahuje požadavky a kritéria pro hodnocení projektu. Důkladně si jej nastudujte a postupujte podle něj. Na konci dokumentu se pro přehlednost nachází checklist, pomocí kterého si prosím zkontrolujte, že odevzdávaný archiv má všechny formální náležitosti.
- Archiv `kry_proj2_template.zip` obsahuje šablonu pro implementaci projektu:

- Soubor `ec.py` pro operace na eliptických křivkách potřebných pro ECDH.
- Soubor `client.py` pro implementaci klienta pro komunikaci a výměnu klíčů.
- Soubor `server.py` pro implementaci serveru pro komunikaci a výměnu klíčů.
- Složku `example_keys`, která obsahuje vzorové výstupní soubory – klíče klienta a serveru – pro všechny varianty výměny klíčů:
  * `client.priv`, `server.priv`, `client.pub`, `server.pub` – soukromé a veřejné klíče klienta a serveru pro klasický Diffie-Hellman algoritmus.
  * `client.shared`, `server.shared` – vypočtený sdílený klíč po dokončení klasické DH výměny.
  * `client_ec.priv`, `server_ec.priv`, `client_ec.pub`, `server_ec.pub` – soukromé a veřejné klíče klienta a serveru pro ECDH.
  * `client_ec.shared`, `server_ec.shared` – vypočtený sdílený klíč po dokončení ECDH výměny.
- Testovací skript `check_zip.sh`, který můžete použít na kontrolu, že je odevzdávaný archiv formálně v pořádku – tedy obsahuje potřebné soubory, je správně pojmenovaný, má správný formát apod.

Kromě těchto materiálů vycházejte z informací z přednášek, dostupných prezentací a dalších materiálů kurzu KRY. V rámci projektu se očekává aktivní studium souvisejících témat, zejména kryptografie založené na eliptických křivkách a algoritmů DH a ECDH pro výměnu klíčů. Můžete také vycházet z (poměrně hutného, ale moderního) standardu NIST SP 800-56A Rev. 3[1].

2 Specifikace projektu

Cílem tohoto projektu je implementace a porovnání dvou variant výměny klíčů: klasického Diffie-Hellmanova algoritmu (DH) a jeho varianty založené na eliptických křivkách (ECDH). V rámci projektu:

1. Do souboru `ec.py` implementujte základní operace nad eliptickými křivkami (sčítání bodů, násobení bodu skalárem, generování klíčů) do předem připravených metod.
2. Použijte implementované operace pro výměnu klíčů pomocí ECDH. Komunikaci implementujte do souborů `client.py` a `server.py` pro odpovídající klientskou a serverovou část komunikace.
3. Pro porovnání implementujte klasický Diffie-Hellmanův algoritmus pro výměnu klíčů. Implementujte opět do souborů `client.py` a `server.py`, stejně jako pro ECDH.
4. Simulujte výměnu klíčů přes nezabezpečený kanál pomocí DH a ECDH (generování klíčů, výměna, výpočet sdíleného klíče).
5. Porovnejte výkon, rychlost, bezpečnost a případně další vlastnosti DH a ECDH.
6. Sepište dokumentaci o **maximální délce tři strany formátu A4**. Dokumentace bude obsahovat stručný popis řešení projektu a popis vaší implementace, dále provedené experimenty a jejich vyhodnocení. Zaměřte se zejména na porovnání rychlosti a diskuzi o velikosti klíčů k dosažení ekvivalentní úrovně bezpečnosti[2].

[1] https://csrc.nist.gov/pubs/sp/800/56/a/r3/final

[2] Nad rámec projektu můžete v dokumentaci diskutovat i porovnání bezpečnosti vzhledem ke kvantovému počítání a postkvantové kryptografii. Vynikající analýza a zpracování na toto téma bude oceněno bonusovými body.

Upozornění

POZOR! Projekt řešte samostatně, jakékoliv opisování povede k disciplinární komisi a k hodnocení 0 bodů. To platí i pro používání generativních nástrojů jako jsou GitHub Copilot, ChatGPT apod. I nekompletní řešení se hodnotí kladně.

3 Technické požadavky

Projekt implementujte v jazyku Python. Referenčním strojem je školní server Merlin a **čistý** Python v3.8. **V rámci projektu je povoleno používat pouze Python standardní knihovnu[3], použití jiných (externích) modulů je zakázáno.** Referenční prostředí si můžete (např. na vlastním stroji) jednoduše vytvořit a aktivovat pomocí nástroje `conda`[4] následujícími příkazy:

Vytvoření referenčního prostředí

```
conda create --name kry_proj2 python=3.8 --no-default-packages
conda activate kry_proj2
```

3.1 Veřejné hodnoty

Pro jednoduchost používejte v projektu běžně používané veřejné parametry pro DH a ECDH. V případě ECDH se jedná o standardizovanou křivku *secp256r1*[5], jejíž hodnoty jsou následující:

Parametry secp256r1 (P-256)

```
p = 0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff,
a = 0xffffffff00000001000000000000000000000000fffffffffffffffffffffffc,
b = 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b,
G = (0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296,
0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5),
n = 0xffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632551,
h = 0x01
```

Pro klasický DH použijte následující veřejný modulus a základ:

Veřejné parametry DH

```
p = 0xffffffffffffffffc90fdaa22168c234c4c6628b80dc1cd129024e088a67cc74020bbea63b
139b22514a08798e3404ddef9519b3cd3a431b302b0a6df25f14374fe1356d6d51c245e485b57662
5e7ec6f44c42e9a637ed6b0bff5cb6f406b7edee386bfb5a899fa5ae9f24117c4b1fe649286651ec
e45b3dc2007cb8a163bf0598da48361c55d39a69163fa8fd24cf5f83655d23dca3ad961c62f35620
8552bb9ed529077096966d670c354e4abc9804f1746c08ca18217c32905e462e36ce3be39e772c18
0e86039b2783a2ec07a28fb5c55df06f4c52c9de2bcbf6955817183995497cea956ae515d2261898
fa051015728e5a8aaac42dad33170d04507a33a85521abdf1cba64ecfb850458dbef0a8aea71575d
060c7db3970f85a6e1e4c7abf5ae8cdb0933d71e8c94e04a25619dcee3d2261ad2ee6bf12ffa06d9
8a0864d87602733ec86a64521f2b18177b200cbbe117577a615d6c770988c0bad946e208e24fa074
e5ab3143db5bfce0fd108e4b82d120a92108011a723c12a787e6d788719a10bdba5b2699c327186a
f4e23c1a946834b6150bda2583e9ca2ad44ce8dbbbc2db04de8ef92e8efc141fbecaa6287c59474e
6bc05d99b2964fa090c3a2233ba186515be7ed1f612970cee2d7afb81bdd762170481cd0069127d5
b05aa993b4ea988d8fddc186ffb7dc90a6c08f4df435c934063199ffffffffffffffff
g = 2
```

[3] https://docs.python.org/3/library/index.html
[4] https://anaconda.org/anaconda/conda
[5] https://neuromancer.sk/std/secg/secp256r1

3.2 Podrobnosti implementace jednotlivých částí projektu

1. `ec.py`

   - Již definované názvy metod prosím kvůli automatickým testům neměňte. V souboru je také nachystána incializace *secp256r1*. Tuto inicializaci (včetně jména proměnné inicializované křivky `curve_secp256r1`) prosím také neměňte. Změna struktury a názvů v souboru povede ke spadnutí automatických testů a tedy hodnocení 0 bodů.

2. `server.py` a `client.py`

   - Server i klient jsou spustitelné skripty podporující výměnu klíčů pomocí DH a ECDH. Komunikace musí probíhat přes síťový protokol (např. `sockets`). Výměna probíhá lokálně na `localhost`. Konfiguraci serveru i klienta určují dva argumenty:
     (a) Argument `--port` určuje, na kterém portu bude komunikace probíhat. Hodnoty jsou z rozsahu 1024-65535.
     (b) Přepínač `--ec` spustí server či klient v režimu ECDH. Bez přepínače proběhne výměna pomocí klasického DH.

     Příklady spuštění

     Spuštění serveru pro ECDH výměnu naslouchajícího na portu 12345:
     `python3 server.py --ec --port 12345`

     Spuštění klienta pro klasickou DH výměnu na portu 7734:
     `python3 client.py --port 7734`

   - Protokol komunikace je následující:
     1. Nejprve je spuštěn server, který naslouchá na zadaném portu.
     2. Následně je spuštěn klient, který se na zadaném portu připojí k serveru.
     3. Po navázání spojení proběhne výměna klíčů (podle (ne)zadaného argumentu `--ec` buď ECDH nebo klasický DH).
     4. Po výměně proběhne výpočet sdíleného klíče a klient i server se ukončí.
   - **Výsledný sdílený klíč je SHA-256 hash sdílené hodnoty** – v případě ECDH tedy x-ové (první) souřadnice sdíleného bodu, v případě klasického DH tedy hexadecimální zápis (bez předpony '`0x`'!) výsledku modulárních operací.
   - Klient i server ukládají vytvořené klíče do souborů v kořenovém adresáři projektu. Názvy a formáty souborů {`client|server`}.{`priv|pub|shared`} jsou důležité pro automatické testování projektu; nedodržení konvence povede ke spadnutí automatických testů a tedy hodnocení 0 bodů. Soubory mají název dle komunikující strany a typu klíče. Soubory se jmenují vždy stejně, nezávisle na použitém algoritmu výměny, tedy vždy `server.priv` pro privátní klíč serveru, `client.pub` pro veřejný klíč klienta apod. Klíče ukládejte v následujícím formátu (viz vzorové klíče):
     – Privátní klíče vždy jako dekadický zápis celého čísla.
     – Veřejné klíče jako dekadický zápis celého čísla v případě klasické DH výměny, v případě ECDH uložte bod jako slovník, ve kterém 'x' a 'y' označují první, resp. druhou souřadnici bodu; hodnotu opět v dekadickém zápisu[6].
     – Sdílené klíče jako SHA-256 hash ve formátu (hex)digest[7].

V rámci projektu můžete samozřejmě vytvářet další soubory a funkce potřebné pro implementaci; nezapomeňte je následně zabalit do odevzdávaného archivu.

[6] Např. {'x': 623737953644898698934082285578232036696877396256571212745988544341551861117 0, 'y': 384334103900650431133821629273691425151569404313390546603896253029616751430 34}

[7] Např. `f2ad1a2daad065ddba850d3e3ab87444ffc7abb1eec7a45532de69ce97c90240`, viz vzorové klíče.

4 Odevzdání a hodnocení

Do e-learningu odevzdávejte archiv pojmenovaný vaším VUT číslem (např. `123456.zip`) ve formátu `zip`. Archiv bude obsahovat řešení projektu, tedy vaši implementaci a `pdf` dokumentaci pojmenovanou vaším VUT číslem (např. `123456.pdf`). Nedodržení konvence pojmenování bude postiženo bodovou srážkou. Pro formální kontrolu můžete použít přiložený skript `check_zip.sh`. **Deadline pro odevzdání projektu je 27. 4. 2025 23:59**.

Za implementaci potřebných operací na eliptických křivkách v souboru `ec.py` lze získat až tři body. Jedná se o správnou implementaci metod `point_add`, `point_mul` a `generate_keys`. Za správnou implementaci DH a ECDH výměny klíčů včetně klient-server komunikace lze získat maximálně čtyři body. Řešení bude testováno proti referenční implementaci i společně (tedy komunikace vašeho serveru a vašeho klienta).

Za porovnání DH a ECDH a dokumentaci je možné získat až tři body. Bude hodnocen popis implementace, porovnání DH a ECDH. Dokumentaci pište raději kratší, ale výstižnou, než dlouhé obecné texty. Zaměřte se zejména na svoji práci. Nemusíte tedy psát úvod do kryptografie eliptických křivek, zaměřte se na vámi implementované operace, způsob generování klíčů, bezpečnost implementované klient-server komunikace, DH a ECDH, jejich slabiny a silné stránky.

5 Checklist před odevzdáním

- ☐ Řešení nic nevypisuje na standardní výstup (žádné debug výpisy).
- ☐ Klient a server všechny použité klíče ukládají do správných souborů. V případě kompletní implementace jsou obsahy souborů `client.shared` a `server.shared` shodné.
- ☐ Implementované řešení funguje bez externích modulů a knihoven. Lze jej spustit na serveru Merlin nebo v referenčním prázdném `conda` prostředí.
- ☐ V odevzdaném archivu se nacházejí soubory `ec.py`, `server.py`, `client.py`, případně další potřebné zdrojové soubory pro spuštění projektu.
- ☐ V odevzdaném archivu se nachází dokumentace pojmenovaná vaším VUT číslem ve formátu `pdf`.
- ☐ Odevzdaný archiv je pojmenovaný vaším VUT číslem a má formát `zip`. Kontrola pomocí skriptu `check_zip.sh` nehlásí žádný problém.

Přílohy

A Lokální instalace `conda`

Na serveru Merlin bohužel není systémový balík `conda` dostupný. Můžete využívat výchozí Python prostředí, které je na Merlinovi dostupné příkazem `python3` a instalaci `conda` vůbec nemusíte řešit.

Pokud byste potřebovali na Merlinovi nebo na svém (Linux) stroji vytvořit lokální instalaci `conda`, můžete využít následující příkazy:

Lokální instalace `conda`

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
~/miniconda3/bin/conda init
source ~/.bashrc
```

Během instalace musíte potvrdit souhlas s licenčními podmínkami a potvrdit některá nastavení, ale jinak se jedná o velmi jednoduchý proces. Po instalaci můžete využít příkazy ze sekce 3 k vytvoření referenčního prostředí pro řešení projektu.

B Recept na jednoduchou hrníčkovou bábovku

Části této přílohy byly vygenerovány pomocí ChatGPT.

Kryptografie je tvrdý oříšek – eliptické křivky, bezpečnostní analýza, správná implementace... To vše dá člověku zabrat. Ale co když se místo lámání šifer chcete na chvíli odreagovat a chcete *rozlomit* jen pár vajec? Proto vám přinášíme recept na hrníčkovou bábovku!

Ať už jste právě dokončili implementaci projektu, nebo jste si uvědomili, že deadline byl včera, není nic lepšího, než si osladit den. Bábovka sice nešifruje data, ale rozhodně zašifruje vaše chuťové buňky do stavu euforie. Takže, pokud vám nevychází testování, můžete si upéct něco, co vyjde vždycky.

B.1 Potřebné suroviny

- 2 hrnky hladké nebo polohrubé mouky
- 1 hrnek mléka
- 1/2 hrnku cukru (dle chuti klidně více či méně)
- 1/2 hrnku oleje
- 1 prášek do pečiva
- 1 vanilkový cukr
- 2 vejce
- tuk na vymazáni formy
- trochu mouky na vysypání vymazané formy
- co nejvíce lásky

B.2 Postup přípravy

1. Bábovkovou formu vymažeme tukem a vysypeme moukou.
2. Všechny suroviny smícháme dohromady. Těsto metličkou dobře promícháme, aby v něm nebyly žmolky mouky.
3. Do těsta můžeme přidat *speciální přísadu*, která bábovku ozvláštní. Osobně mám nejraději skořici, ale lze použít i kakao, kandované nebo nakrájené čerstvé ovoce, najemno nakrájené vlašské ořechy, čokoládu nebo podobné pochutiny.
4. Těsto vylijeme do formy.
5. Formu s těstem vložíme do trouby a pečeme cca 45-50 minut na 180°C. Propečenost vyzkoušíme vpichem špejle – dokud zůstává mokrá, je potřeba ještě chvíli péct; pokud zůstane suchá, máme upečeno.
6. Po zchladnutí vychutnáváme lahodnou chuť bábovky. V případě méně povedeného výsledku můžeme bábovku sdílet a *vychutnávat* s kamarády a blízkými.