# Generátor úloh do aplikované kryptografie
# Kontrolní studie

Michal Homola,
Dominik Chrenčík,
Jiří Marák,
Vojtěch Lukáš

26. března 2023

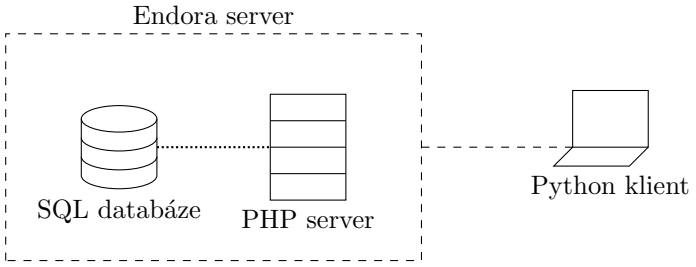
# Obsah



# Úvod

# 1 Architektura

Schéma připravovaného systému lze vidět na obr. 1. Úlohy budou uloženy v SQL databázi. K této databázi bude mít přístup pouze webový PHP server. Ten slouží jako „prostředník“ mezi klientem a databází. Dále by měl do úloh vkládat náhodná data (klíče apod.) a případně také vyhodnocovat výsledky. Klientská aplikace bude fungovat jako přístupový bod a sehrávat roli prezentační vrstvy. Pro jednoduchost bude vyvinuta v jazyce Python s oddělenou logickou vrstvou. Bude tedy možné na tuto vrstvu napojit i jednoduché grafické rozhraní.




Obrázek 1: Schéma systému

Tabulka 1: Struktura SQL databáze

| **ID**<br>`INT` | **Zadání**<br>`TEXT` | **Kód**<br>`VARCHAR(5)` | **Nápověda**<br>`TEXT` |
|---|---|---|---|
| 1 | Rozhodněte (ano/ne) zda je číslo $n =$ \$1 prvočíslo | `PR` | ... |
| 2 | Zašifrujte zprávu $m =$ \$4, pomocí RSA kryptosystému. Prvočísla jsou $p =$ \$1; $q =$ \$2, a soukromý klíč je $e =$ \$3 | `RSAe` | ... |
| ⋮ | ⋮ | ⋮ | ⋮ |

## 1.1 Databáze úloh

V rámci návrhu kryptografických úloh, byly úlohy co nejvíce standardizované, z důvodu jejich uložení v jednotné databázi. Zároveň musí být výsledky jednoznačné tak, aby byly dobře porovnatelné s výsledky zadanými řešitelem. Proto jsou výsledky koncipovány jako číslo, či prosté „ano/ne“.

V databázi jsou zahrnuty různé typy kryptografických úloh, aby bylo možné si vyzkoušet různé druhy kryptografických algoritmů a technologií. Součástí každé úlohy bude i nápověda k řešení, tak aby i řešitel, který si neví rady, mohl příklad vyřešit.

## 1.2 Konstrukce databáze

V tabulce 1 lze vidět současnou strukturu SQL databáze. Sloupec **ID** slouží jako primární klíč databáze, v buňce **Zadání** se pak nachází textový popis úlohy. Zde stojí za povšimnutí, že všechny číselné hodnoty důležité k výpočtu jsou nahrazeny zástupnými znaky $\$n$. Na místa těchto znaků bude logika v back-endu vkládat vygenerované hodnoty. Díky tomu bude možno jednu úlohu řešit vícekrát, pokaždé s jinými parametry. **Kód** úlohy pak slouží pro snazší rozlišení úloh. Uživatel si bude moct vybrat jaký typ bude chtít řešit, back-end si tuto úlohu vytáhne z databáze, opatří ji vygenerovanými operandy a spolu se správným výsledkem a nápovědou z buňky **Nápověda** ji zašle uživateli.

### 1.2.1 RSA

Část úloh zahrnuje výpočet RSA algoritmu. Zadáno je zde šifrování nebo naopak dešifrování zprávy pomocí zadaného klíče algoritmem RSA.

Vygenerování klíče:

$$n = p \cdot q$$
$$\phi(n) = (p-1) \cdot (q-1)$$
$$e, d = \text{pro které platí } (d \cdot e) \bmod \phi(n) = 1$$

Veřejný klíč: $(n, e)$

Soukromý klíč: $d$

Šifrování:

Zpráva $M$

$$C = M^e \bmod n$$

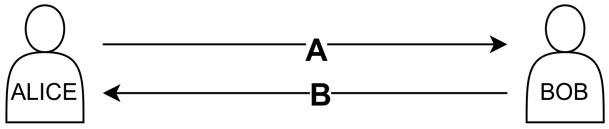
Dešifrování:

Kryptogram: $C$

$$M = C^d \bmod n$$

### 1.2.2 Diffie-Hellman

Další úloha je použití protokolu Diffie-Hellman, který slouží k výměně klíčů mezi dvěma stranami bez potřeby sdílení klíčů předem. Protokol je založen na principech asymetrické kryptografie a umožňuje ustanovení šifrovacího klíče přes nedůvěryhodné médium. V úloze se ověřuje pochopení procesu výměny a výpočtu klíčů mezi dvěma stranami.




Obrázek 2: Diffie-Hellman - předání klíčů

| ALICE | BOB |
|---|---|
| $a \in_R Z_q$ | $b \in_R Z_q$ |
| $A = g^a mod\, p$ | $B = g^b mod\, p$ |
| $K = B^a mod\, p$ | $K = A^b mod\, p$ |

### 1.2.3 Test prvočíselnosti

Test prvočíselnosti se také nachází v databázi úloh. Tato úloha zahrnuje ověření, zda je zadané číslo prvočíslo či nikoliv, což je základem pro mnoho kryptografických algoritmů. Pro testovaní zda je číslo prvočíslem lze použít například

Lucas–Lehmerův test prvočíselnosti pro Mersennova prvočísla:

Nechť $p$ je liché prvočíslo a $M = 2^p - 1$
Nastavíme $s_0 = 4$
Pro $i = 1, 2, \ldots, p-2$ :
$s_i = (s_{i-1}^2 - 2) \bmod M$
Pokud $s_{p-2} = 0$, pak $M$ je Mersennovo prvočíslo.
Jinak $M$ je složené číslo.

### 1.2.4 GCD a LCM

Dále jsou zde také příklady na výpočet nejmenšího společného násobku (Least Common Multiple – LCM) a největšího společného dělitele (GCD – Greatest common divisor). které také najdou uplatnění v kryptografii. Při řešení těchto příkladů lze využít například rozklad na prvočísla.

### 1.2.5 Modulární aritmetika

V databázi jsou také příklady na modulární aritmetiku, jako výpočet operace modulo nebo kongruence. Operace modulo ($mod\, n$) vrátí pouze zbytek vstupního čísla po dělení číslem $n$.

Cele číslo je kongruentní modulo $n$ s celým číslem $b$ tehdy, když je rozdíl $a-b$ dělitelný číslem $n$. Zapisujeme jako $a \equiv b \pmod{n}$. Pro kongruence modulo platí:

$$a \equiv b \pmod{m}$$
$$b \equiv a \pmod{m}$$

V dalším vývoji by mohla být nápověda rozdělena na více úrovní, tak aby si je řešitel mohl postupně vyžádat v případě, že mu bude postačovat pouze malá nápověda a potřebuje pouze „nakopnout“, nebo si neví rady a potřebuje nápovědu větší. Taktéž budu přidány další úlohy z oblasti aplikované kryptografie.

## 1.3 Back-end

Architektura back-endu je navržena podle doporučení REST API. Původní návrhy řešení počítaly s využitím .NET serveru na portálu Microsoft Azure. Nakonec bylo ale upřednostněno řešení využívající PHP server. Celé řešení back-endu bylo založeno na [1]. Od začátku byl projekt vyvíjen přímo na serveru pro usnadnění přístupu.

Tabulka 2: Tabulka API dotazů

| URL | Popis |
|---|---|
| `/index.php/alltasks` | Vrátí JSON objekt obsahující všechny úlohy z databáze |
| `/index.php/task?code=<kód>` | Vrátí jednu (nebo více) úloh(u) daného typu |
| `/index.php/randomtask` | Vrátí jednu náhodně vybranou úlohu |

### 1.3.1 Soubory webového serveru

```
web
├── Model
│   ├── Database.php
│   └── TaskModel.php
├── Controller
│   └── Api
│       ├── BaseController.php
│       └── TaskController.php
├── inc
│   ├── config.php
│   └── bootstrap.php
└── index.php
```

Moduly ve složce `Model` slouží k propojení a komunikaci s databází. Modul `TaskController.php` je využit ke zpracování různých požadavků z uživatelské aplikace. Soubory ve složce `inc` by se daly označit jako „servisní moduly“: `config.php` definuje přístupové údaje k databázi jako konstanty[1]; modul `bootstrap.php` pak obstarává správné propojení modulů. Modul `index.php` je pak ten, který přímo komunikuje s klientskou aplikací, třídí její požadavky a korektně na ně odpovídá.

## 1.4 API dotazy

V rámci projektu budou využity dotazy z tabulky 2. Struktura těchto dotazů respektuje doporučení REST API. Demonstrační příklad je možné navštívit na tomto odkazu.

[1] Vzhledem k citlivé povaze obsažených informací je tento soubor vyřazen z verzování pomocí záznamu v `.gitignore`.

### 1.5 Front-end

## 2 Současný stav

## Závěr

## Reference

[1] SONI, Sajal. How to build a simple REST API in PHP. *Envato Tuts+* [online]. 27-5-2021 [cit. 2023-03-25]. Dostupné z: `https://code.tutsplus.com/tutorials/how-to-build-a-simple-rest-api-in-php--cms-37000`