Generátor úloh do aplikované kryptografie
Kontrolní studie

Michal Homola,
Dominik Chrenčík,
Jiří Marák,
Vojtěch Lukáš

26. března 2023

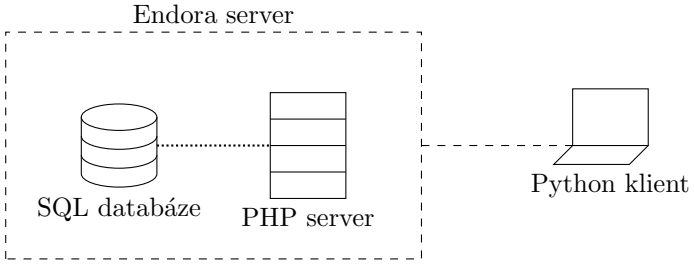
Obsah

Úvod

Předmětem této studie je představit vizi projektu s názvem „Generátor kryptografických úloh“. V první části bude naznačena architektura systému, ve stručnosti rozebrány jednotlivé typy úloh, popsána struktura databáze, front-end a back-end. Druhá část na tyto návrhy naváže a upřesní současný stav vývoje. V závěru jsou pak představeni členové týmu a jejich příspěvek.

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

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.

1.1.1 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 Typy úloh

1.2.1 RSA

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

Tabulka 1: Struktura SQL databáze

ID `INT`	**Zadání** `TEXT`	**Kód** `VARCHAR(5)`	**Nápověda** `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`	...
⋮	⋮	⋮	⋮

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:

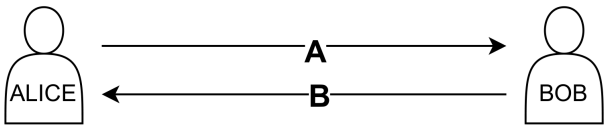
Kryptogram: C

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

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.

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



Obrázek 2: Diffie-Hellman - předání klíčů

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 je založeno na [1]. Od začátku byl projekt vyvíjen přímo na serveru pro usnadnění přístupu.

1.4 Výpočet řešení úloh

1.4.1 API dotazy

V rámci projektu jsou využity dotazy z tabulky 2. Struktura těchto dotazů respektuje doporučení REST API.

1.4.2 Generování hodnot

Úkolem back-end serveru bude také generování a vkládání hodnot do textu úlohy. Toto bude mít na starosti samostatný modul, který podle kódu vyžádané úlohy vygeneruje potřebné hodnoty, vloží je na místo tokenů „$$n$“ (viz tabulka 1). Takto připravená data již může převzít další modul a odeslat je uživateli.

Neméně důležitým úkolem této sekce je také výpočet správného výsledku pro vyhodnocení. Správný algoritmus pro výpočet bude zvolen podle kódu úlohy. Vypočtená hodnota bude opět předána dalšímu modulu, který je bude odesílat uživateli.

1.4.3 Obsah JSON odpovědi

Jak již bylo zmíněno v předchozích odstavcích, JSON odpovědi budou obsahovat kromě buněk z SQL databáze také další hodnoty vygenerované serverem. V tuto chvíli vypadá návrh JSON objektu následovně:

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

- název úkolu,
- zadání úkolu,
- kód úkolu,
- nápověda,
- správné řešení.

1.5 Front-end

Tato část projektu bude mít formu jednoduché aplikace vyvinuté v jazyce Python. Nejprve si vyžádá seznam úloh, které jsou uživateli k dispozici. Uživatel si z výběru zvolí jednu úlohu, ta je mu pak prezentována. V tuto chvíli má uživatel také možnost vyžádat si nápovědu. Po zadání řešení je pak vyhodnocena správnost výsledku. Nyní se nabízí možnost počítat skóre a čas potřebný k vyřešení.

Aplikace může mít více podob. Implicitně se počítá s konzolovou aplikací, ale nic nebrání ani vývoji elementárního grafického rozhraní pro příjemnější uživatelský zážitek.

2 Současný stav

Tým využívá dva GIT repozitáře. K nahlédnutí jsou zde a zde.

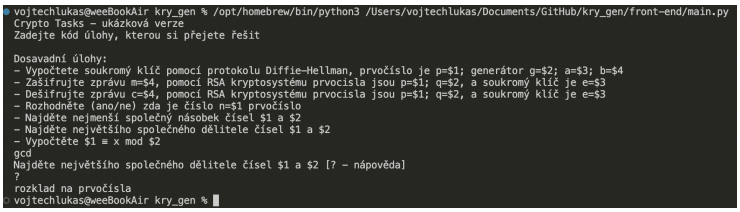
2.1 Databáze úloh

V čase odevzdání této studie je navrhnuto 7 různých kryptografických úloh v souladu s jejich návrhem v sekci 1.1. Všechny tyto úlohy jsou opatřeny tokeny „$\$n$“ a nápovědami. Jsou také již nahrané v SQL databázi na serveru.

2.2 Back-end

Webová služba je v tuto chvíli zčásti hotová a publikovaná na adrese `http://vut-fekt-mpckry-gr14.8u.cz/index.php`. Implementováno jsou nyní reakce na API dotazy (viz sekce 1.4.1 a 1.4.3), chybí implementovat generování hodnot (viz sekce 1.4.2). V textu JSON odpovědí jsou tokeny „$\$n$“ nyní ještě nenahrazeny. Odpověď nyní také ještě neobsahuje hodnotu se správným výsledkem.

Funkčnost lze ukázat na následujícím odkazu, který respektuje API popsané v tabulce 2: `http://vut-fekt-mpckry-gr14.8u.cz/index.php/task?code=dh`. Po kliknutí na odkaz by se měl uživateli zobrazit v prohlížeči JSON objekt obsahující úlohu týkající se DH protokolu.



Obrázek 3: Ukázkový průchod front-end skriptem

2.3 Front-end

Této části systému bylo prozatím věnováno nejméně pozornosti. Byl vyvinut ukázkový skript, který dokazuje úspěšné propojení s webovou službou, zobrazení všech úloh, žádost a přijetí jedné konkrétní úlohy a výpis nápovědy. To vše lze vidět na obr. 3

3 Členové týmu

Michal Homola

Má na starosti vývoj logiky generování hodnot a výpočtu správného výsledku.

Dominik Chrenčík

Vyvíjí front-end aplikace.

Jiří Marák

Vymýšlí kryptografické úlohy a nápovědy k nim.

Vojtěch Lukáš

Má na starosti back-end a API.

Závěr

Během dosavadní činnosti na projektu byla navržena architektura systému, schéma databáze, podoba API a JSON odpovědí. Bylo také vymyšleno několik kryptografických úloh, vyvinut back-end a elementární front-end. Všechny komponenty mezi sebou úspěšně komunikují.

V dalších fázích projektu je třeba zaměřit se na logiku generování hodnot do jednotlivých úloh a na zlepšení uživatelského zážitku při využívání front-endové aplikace.

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`