


Artemis 6.4.0

Course Overview

  ge64baw

Courses > Praktikum: Grundlagen der Programmierung WS22/23 > Exercises > W13H02 - Kryptographuine

W13H02 - Kryptographuine

Hausaufgabe

Medium

Submission due:

6 months ago

Complaint due:

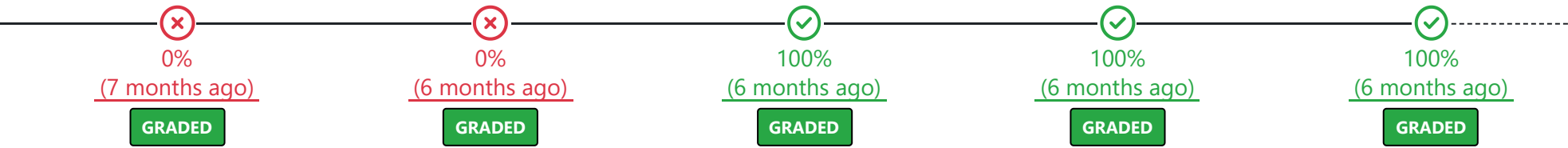
6 months ago

Points: 6 of 6

Assessment: automatic ?

 100% (6 months ago) GRADED

Recent results:



Show all results

Tasks:

Kryptographuine

Die Forschuine der PUM gehen in ihrem Alltag auch wichtiger Forschungsarbeit nach. Dabei werden oft sensible Daten von Versuchuinen aufgenommen, welche sie ab und zu mit anderen Fakultäten teilen müssen. Da ihnen Datenschutz sehr am Herzen liegt, würden sie die Daten für den Transfer gerne verschlüsseln. Sie haben sich auch schon für das RSA Verfahren entschieden. Aber leider ist ihnen jüngst ihr MiniJava-Compiler kaputtgegangen. Sie können daher nicht automatisch den benötigten Bytecode generieren und bitten Dich um Hilfe!

Update 26.1 21 Uhr

- Die Kommentare im Template für `cyption` hatten eine andere Reihenfolge der Variablen spezifiziert als Artemis. Die richtige Version ist Artemis. Für alle die, die Aufgabe nach diesem Zeitpunkt begonnen haben, ist dies nicht relevant.

RSA (nicht relevant für das Bearbeiten der Aufgabe)

Das RSA (Rivest–Shamir–Adleman) ist ein kryptografisches Verfahren, welches heutzutage in vielen Bereichen eingesetzt wird. RSA ist ein sogenanntes Public-Key Verfahren, das heißt es gibt für jede Identität einen öffentlichen Schlüssel, welcher allen Teilnehmern bekannt ist und mit welchem Daten verschlüsselt werden können, so wie einen privaten Schlüssel, welcher nur dem Empfänger bekannt ist und mit welchem wiederum entschlüsselt wird. Um ein Paar aus öffentlichem und privatem Schlüssel zu erstellen, geht man wie folgt vor:

- Generiere zufällig zwei sehr große Primzahlen p und q , $p \neq q$.
- Berechnet $N = pq$.
- Berechne $\lambda(N) = \text{kgV}(p - 1, q - 1)$.
- Wähle zufällig e aus $\mathbb{Z}, \dots, \lambda(N) - 1$, wobei gelten muss $\text{ggT}(e, \lambda(N)) = 1$.
- Berechne d mit $(e \cdot d) \bmod \lambda(N) = 1$.
- e , N sind öffentlich d , $\lambda(N)$, p , q sind geheim.

Eine Nachricht m verschlüsselt man dann mit $m^e \bmod N$ zu einem Ciphertext c , welcher dann mithilfe von $c^d \bmod N$ wieder entschlüsselt werden kann.

Aufgaben

Erweiterter Euklidischer Algorithmus

Die Forschuine können sich - dank Deiner Hilfe in den P-Aufgaben - Primzahlen generieren und den zweiten, dritten und vierten Schritt des RSA-Verfahrens aufgrund Deiner Vorarbeit in der ersten Hausaufgabe auch umsetzen. Jetzt würden sie gerne den fünften Schritt zur Ermittlung von d implementieren.

Hier findet ihr ein Beispielprogramm, welches den Algorithmus berechnet, dieses könnt ihr wie in der ersten Hausaufgabe einfach frei übersetzen oder Euch mit den nachfolgenden mathematischen Erleuterungen einen eigenen Ansatz überlegen (dieser sollte allerdings nicht deutlich ineffizienter sein als der erweiterte euklidische Algorithmus. Einfach alle Möglichkeiten für d auszuprobieren, ist z.B. viel zu langsam und wird mit 0 Punkten bewertet).

```

int rPrior, r, sPrior, s, tPrior, t, q, tmp;

rPrior = read(); // this is the input e
r = read(); // this is the input /\(N)
sPrior = 1;
s = 0;
tPrior = 0;
t = 1;

while (r != 0) {
    q = rPrior / r;
    tmp = r;

    r = rPrior - q * r;
    rPrior = tmp;
    tmp = s;

    s = sPrior - q * s;
    sPrior = tmp;
    tmp = t;

    t = tPrior - q * t;
    tPrior = tmp;
}

write(sPrior); // this is the wanted d

```

Mathematischer Hintergrund - Optional!

In der Praxis bietet sich zur Berechnung von d der erweiterte euklidische Algorithmus $eea(a, b)$ an. Dieser berechnet für zwei Zahlen a und b die Zahlen x und y mit $\text{ggT}(a, b) = ax + by$. Wenn man jetzt den Algorithmus mit $eea(e, \lambda(N))$ ausführt, erhält man mit x das gesuchte d . Das liegt daran, dass $(e \cdot d) \bmod \lambda(N) = 1$ ja nichts anderes heißt, als $(e \cdot d) +$ irgendein Vielfaches $\lambda(N) * y$ von $\lambda(N)$, das durch das "mod $\lambda(N)$ " wieder weggekürzt würde $= 1$. 1 ist dabei nach der Bedingung aus 4. der ggT von e und $\lambda(N)$.

Hinweise zur Abgabe:

- Die Abgabe erfolgt in der Datei `eea.jvm`. Wenn der Pfad oder Name der Datei geändert wird, kann dies dazu führen, dass die Tests fehlschlagen!
- In dieser Aufgabe zählt **nur** das Input/Output-Verhalten Eures Programms. Es muss **nicht** wie in der ersten Hausaufgabe genau übersetzt werden.
- Die Tests werden nur prüfen, ob für eingelesene e und $\lambda(N)$ (in dieser Reihenfolge), welche aus zwei positiven Primzahlen p und q generiert wurden, gilt $(e \cdot d) \bmod \lambda(N) = 1$. Du kannst daher davon ausgehen, dass e und $\lambda(N)$ größer gleich 3 sind.
- Die öffentlichen Tests überprüfen nur, ob Eurer Programm nach der Deadline getestet werden kann.

✓ Abgabe enthält Bytecode [1 of 1 tests passing](#)

✓ Abgabe berechnet richtige Ergebnisse [3 of 3 tests passing](#)

Ver- und Entschlüsselung

Da die Pinguine nun private Schlüssel (bestehend aus $p, q, \lambda(N), d$) und öffentliche Schlüssel (bestehend aus e, N) generieren können, wollen sie diese natürlich auch einsetzen. Sie wollen aber nicht immer ihre Schlüssel aufs Neue eintippen, wenn sie einen großen Datensatz mit vielen kleinen Einträgen verschicken. Zur Erinnerung, die Funktion $f(x, y, N) = x^y \bmod N$ berechnet die Ver- und Entschlüsselung (wobei x und y für m und e bei der Verschlüsselung stehen für c und d bei der Entschlüsselung). Du sollst ihnen daher nun ein Programm in Bytecode implementieren, das wie folgt funktioniert:

1. Lies die Werte y, N (in dieser Reihenfolge) mit `READ` ein.
2. Lies einen neuen Wert x ein, welcher verschlüsselt werden soll.
3. Ist dieser 0 beende das Programm. Sonst berechne $x^y \bmod N$ und gib das Ergebnis mit `WRITE` aus.
4. Fahre mit Schritt 2. fort.

Das Programm soll also einmal vom Nutzer öffentliche/private Schlüssel einlesen und dann solange Nachrichten einlesen und diese mit den Schlüssel ver-/entschlüsseln, bis einmal als Nachricht 0 eingegeben wird.

ABER ACHTUNG: Viele Pinguine nutzen aus Umweltgründen noch alte Geräte, welche den `MOD` Operator noch nicht unterstützen, dieser darf daher in dieser Teilaufgabe NICHT verwendet werden!

Hinweise zur Abgabe:

- Du kannst davon ausgehen, dass y und N größer gleich 1 sind und x größer gleich 0. Negative Zahlen müssen also nicht behandelt werden.
- Die Abgabe erfolgt in der Datei `cryption.jvm`. Wenn der Pfad oder Name der Datei geändert wird, kann dies dazu führen, dass die Tests fehlschlagen!
- In dieser Aufgabe zählt **nur** das Input/Output-Verhalten Eures Programms. Es muss **nicht** wie in der ersten Hausaufgabe genau übersetzt werden.
- Die öffentlichen Tests überprüfen nur, ob Eurer Programm nach der Deadline getestet werden kann.

✓ Abgabe enthält Bytecode [1 of 1 tests passing](#)

✓ Abgabe berechnet richtige Ergebnisse [3 of 3 tests passing](#)

FAQ

Exercise details

Release date:	Jan 26, 2023 18:30
Start date:	Jan 26, 2023 18:30
Submission due:	Feb 12, 2023 18:00
Complaint due:	Feb 19, 2023 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left. 