.Testat VS 2021

.Titel: RSA Primes

.Motivation

Euer Vertrieb hat leider seinen Schlüssel zum Entschlüssel eines Millionenauftrags verloren. Ohne diesen Schlüssel kann er den wichtigen Vertrag nicht entschlüsseln und somit kommt der Auftrag nicht zu Stande. Die Zeit drängt und morgen soll dieser unterschrieben werden. Ihr könnt leider keine neue Verschlüsselte Version anfordern, da dies den Vertrag ebenso gefährdet.
Das einzige was ihr hab, um noch eine Change zu haben, den Vertrag zu entschlüssel ist:

- das Verfahren ist RSA, also muss es ein RSA Schlüssel sein
- ihr habt die Public key
- öffentlicher Exponent und Modulus sind somit bekannt
- bei der Generierung damals wurden nicht alle Primzahlen berücksichtigt (know Bug bei der Schlüsselgenerierung).
-

Mit diesem Wissen müsst ihr nun ein Verteiltes System erstellen um den privaten Schlüssel mittels Brute Force zu ermitteln. Durch den Bug ist die Primzahlenmenge zwar begrenzt, aber dennoch zu hoch um mit nur einem Knoten den Schlüssel zu entschlüsseln.

.Voraussetzung

Als Voraussetzung für dieses Testat dienen die Laborübungen im Fach "Verteilte Systeme". Der Quellcode hieraus kann als Basis für die Rahmenbedingungen verwendet werden, sofern hilfreich.

Für die Entschlüsselung und Schlüsselbestimmung wird Quellcode bereitgestellt.
In diesem Testat liegt der Fokus auf der Verteilung und Performance-Auswertung.
Die Listen der möglichen Primzahlen wird bereit gestellt passen zu den „RUNS“.

.Ziel

Ziel dieser Arbeit ein mit einem verteilten System zu erstellen, um einen privaten Schlüssel aus zwei Primzahlen zu bestimmen. Damit wird dann ein verschlüsselter Text entschlüsselt.

Dabei besteht das System aus 2 Komponenten:

1. Client: der Client übergibt dem Cluster einen public key und den verschlüsselten Text. Der Client erwartet den entschlüsselten Text als Antwort
2. Das Cluster versucht den Schlüssel möglichst schnell zu bestimmen und den Text zu entschlüsseln. Die Zeit, die das Cluster benötigt wird gemessen und daraus erfolgt die Auswertung, wie gut die beiden Primzahlen sind.

Es werden mindestens folgende Kombinationen ausgewertet.

RUN	1	2	3	4
Primzahlen	100 Primzahlen Primzahlen	1 000 Primzahlen	10 000 Primzahlen	100 000 Primzahlen
Knoten /	2	2	2	2

Prozesse				

Die Knotenanzahl wird in zwei weiten Läufen erhöht:

RUN	1	2	3	4
Primzahlen	100 Primzahlen Primzahlen	1 000 Primzahlen	10 000 Primzahlen	100 000 Primzahlen
Knoten / Prozesse	5	5	5	5

RUN	1	2	3	4
Primzahlen	100 Primzahlen Primzahlen	1 000 Primzahlen	10 000 Primzahlen	100 000 Primzahlen
Knoten / Prozesse	10	10	10	10

Da die Berechnung auf mehreren Knoten des Clusters abläuft, muss die Berechnung effizient verteilt werden. Eine Verteilung ist zu definieren und umzusetzen. Kein Knoten rechnet alle Primzahlenkombinationen nur ein Subset.

Das Cluster muss während der Berechnung Ausfälle ausgleichen, sodass eine Berechnung immer alle gegebenen Primzahlen abdeckt.
Ein so entstehender „Ruckler“ muss in der Bewertung berücksichtigt werden.

Ermitteln Sie die Zeiten und machen eine Prognose, wie lange es dauert, um 1 Billionen Primzahlen mit der jeweiligen Clustergröße von 2,5 und 10 zu berechnen.

Zeigen Sie auf, wie groß das Cluster sein müsste, um 1 Billionen Kombinationen in 1h zu berechnen.

HINWEIS: Terminiert die Berechnung von 100 000 Primzahlen nicht in absehbarer Zeit (<2h) so kann die Anzahl auch verringert werden.

.Abgrenzung

Für das Testat sollen folgende Punkte als Grundlage dienen.

.Inhalt

Das Cluster muss während der Laufzeit keinen „Rejoin“ berücksichtigen, lediglich den Ausfall kompensieren.
Die Bibliotheken für die Cryptooperationen werden per JAR und per Github – Projekt bereitgestellt.

Die Listen der möglichen Primzahlen wird bereit gestellt passen zu den „RUNS“.

.Betriebssystem und Laufzeitumgebung

Das Betriebssystem ist freigestellt, jedoch soll die Anwendung auf einem Raspberry PI (3B) mit einer JRE Laufzeitumgebung >= JRE-1.8.0 laufen, die als Abnahmeumgebung zu berücksichtigen ist.
Es muss kein Raspberry PI dafür verwendet oder für die Entwicklung angeschafft werden, jedoch sind die Hardware-Limitationen zu berücksichtigen.
Für die Messung sind die Werte auf der eigenen HW zu berücksichtigen.

.Serverumgebung und Knoten

Die Anwendung soll min. 10 Knoten unterstützen können. Als Knoten zählt eine Knotenanwendung. Jedoch sind die Anwendungen auf min. 2 Rechner verteilt werden. (Abnahme erfolgt auf min. 2 Raspberry Pis).

.weiteres

- Keine UI, Konsolenprogramm mit Startparametern
- der PRIM-Zahlen-Raum sind vorab durch die Messanforderung begrenzt.

.Zeitpunkt

Der Abgabezeitpunkt wird separat bekannt gegeben per:

- Moodle UND per Email

Als Eingangskriterium dient:

MIN (<Zeitstempel des Email – Empfangens>, <Zeitstempel Moodle>) (!)

.Form

Die Abgabe findet in elektronischer Form (Email) statt.
Zusätzlich wird das Abgabesystem (Moodle) verwendet (Kursseite => in der Gruppe).

Die Dateien im Anhang sind gepackt als ZIP-Datei (eine für alle Anhänge) anzufügen.
Im Betreff jeder Email ist folgende Struktur anzugeben:

TESTAT21_<KURS>_<GRUPPE>.zip

z.B.:

TESTAT21_TINF19X_01.zip

In der Abgabe sind die einzelnen Matrikelnummer aufzuführen (als Teil der schriftlichen Ausarbeitung)

.Inhalt

Der Inhalt des Testates umfasst:

Bezeichnung	**Beschreibung**	**Umfang**
Quellcode	der gesamte Quellcode samt verwendeter Ressourcen und Kommentaren	-
Lauffähige Ausführung	Ausführbare Version des Programms samt Startroutine	-
Analyse	Aufgabenanalyse	Max. 1-2 Seiten
Grobkonzept	Konzeptionelle Darstellung des Ablaufes und beschreibender Text (Ablaufplan, Grobarchitektur)	Max. 11-13 Seiten (Kurz halten)
Testplan	Abnahme- und Testplan für die "Abnahme", Protokoll und Ergebnisse.	10 Testfälle reichen aus
Performance-Auswertung	s.o.	1 Auswertung inkl. Text, max. 1 Seite
Fazit	Reflexion über das Ergebnis	1-2 Seiten