ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2016 Prof. R. Wattenhofer

Technische Informatik II

Probeklausur

Donnerstag, 02. Juni 2016, 10:15 – 11:45 Uhr

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten und beschriften Sie Skizzen und Zeichnungen verständlich.

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie jedes beschriebene Blatt mit Ihrem Namen und Ihrer Legi-Nummer.

Name	**Legi-Nr.**

Punkte

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1		15
2		15
3		8
4		7
5		13
6		9
7		13
8		10
Summe		**90**

1 Multiple Choice (15 Punkte)

Aussage	**Wahr**	**Falsch**
Man kann die Entschlüsselung von ECB parallelisieren.	☐	☐
Elgamal-public-key-Verschlüsselung schützt gegen replay-Attacken.	☐	☐
Wenn ein multi-commodity flow network mit den Raten $r_1, \ldots, r_k$ in ein Netwerk passt, dann passt auch ein unsplittable multi-commodity flow mit den selben Raten in das selbe Netzwerk.	☐	☐
Damit eine Hashtabelle gute Laufzeiten bietet, muss die verwendete Hashfunktion collision resistant sein.	☐	☐
Ein outer join mit einer leeren Tabelle ändert die Anzahl der Ergebniszeilen nicht.	☐	☐

2 Hashing (9 Punkte)

a) `[6 Punkte]` Wir betrachten die parametrisierte Hashfuntion $h_i(k) = h(k) + i + i^2 \mod m$ für eine beliebige Hashfuntion $h : U \to [m]$ und eine Primzahl $m > 2$. Zeigen Sie, dass die probing sequence für jedes $k \in [m]$ höchstens $\lceil \frac{m}{2} \rceil$ Elemente von $[m]$ abdeckt.

b) `[3 Punkte]` Sei unser Universum $U = [2^{2w}]$ die Menge aller Bistrings der Länge $2w > 0$, sei unsere Hashtabelle das Array $[2^w]$, und sei $h : U \to [2^w]$ eine Hashfunktion. Zeigen Sie, dass es eine Menge von mindestens 2^w keys gibt, die alle denselben Hash haben.

3 Key Exchange Variante (10 Punkte)

Betrachten Sie die folgende Variante eines Key-Exchange-Algorithmus:

Algorithmus 1: Neues Schlüsselaustauschverfahren mit Forward Secrecy

Input : Alices und Bobs gemeinsamer geheimer Schlüssel k_{shared}.
Ergebnis: Ein Diffie-Hellman Schlüsselaustauschverfahren (key exchange), das hoffentlich nicht gegen eine man-in-the-middle-Attacke anfällig ist und zusätzlich forward secrecy bietet.

1 Alice und Bob führen den Algorithmus Diffie-Hellman Key Exchange aus, um Rundenschlüssel (round key) $g^{k_A k_B}$ zu erhalten; k_A und k_B sind die privaten Exponenten von Alice und Bob
2 Alice und Bob verschlüsseln $g^{k_A k_B}$ mit k_{shared} und senden sich das Ergebnis gegenseitig zu
3 Falls die Entschlüsselung $g^{k_A k_B}$ ergibt, so akzeptieren beide $g^{k_A k_B}$ als den Rundenschlüssel

a) `[2 Punkte]` Inwiefern ist Diffie-Hellman gegen eine man-in-the-middle-Attacke anfällig?

b) `[3 Punkte]` Hat Algorithmus 1 die Eigenschaft forward secrecy?

c) `[5 Punkte]` Ist Algorithmus 1 anfällig gegen eine man-in-the-middle-Attacke?

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2016 Prof. R. Wattenhofer

Technische Informatik II

Klausur

Mittwoch, 24. August 2016, 15:00 - 16:30 Uhr

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten und beschriften Sie Skizzen und Zeichnungen verständlich. Was wir nicht lesen können, können wir auch nicht bewerten!

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie **jedes Zusatzblatt** ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Name	**Legi-Nr.**

Punkte

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1		24
2		12
3		10
4		16
5		12
6		16
Summe		**90**

1 Multiple Choice (24 Punkte)

Geben Sie bei jeder Aussage an, ob sie wahr oder falsch ist. Jede korrekte Antwort gibt 1 Punkt, **jede fehlerhafte Antwort -1 Punkt**. Jede unbeantwortete Aussage gibt 0 Punkte. Wenn die Summe der Punkte für alle Aussagen negativ ist, dann wird die Aufgabe insgesamt mit 0 Punkten bewertet. Sie können Ihre Antworten bei dieser Aufgabe nicht begründen. Anmerkungen und Fallunterscheidungen werden bei der Bewertung ignoriert, nur das tatsächlich angekreuzte Kästchen wird bewertet.

Aussage	**wahr**	**falsch**
In einer max-min-fairen Zuweisung (allocation) haben zwei Flüsse immer die gleiche Rate, wenn sie die gleiche Nachfrage (demand) haben und über mindestens eine gemeinsame Kante verlaufen.	☐	☐
In einer Datenbank legen wir eine Tabelle für jede Entity an, hingegen nie für eine Relationship.	☐	☐
Gegeben zwei Tabellen T_1 und T_2, die eine Grösse (Anzahl der Tupel) von s_1 bzw. s_2 haben, so ist die Grösse von $join(T_1, T_2)$ höchstens $s_1 \times s_2$.	☐	☐
Zwei verschiedene Datenbanken haben immer unterschiedliche ER-Diagramme.	☐	☐
$B+$-trees können nur für Attribute verwendet werden, auf denen eine Ordnung definiert ist.	☐	☐

Lösungen

Aussage	**wahr**	**falsch**
In einer max-min-fairen Zuweisung (allocation) haben zwei Flüsse immer die gleiche Rate, wenn sie die gleiche Nachfrage (demand) haben und über mindestens eine gemeinsame Kante verlaufen. *Begründung: Gegenbeispiel: die Rate des einen Flusses könnte durch eine Kante mit sehr kleiner Kapazität, die der andere Fluss nicht benutzt, beschränkt sein.*		✓
In einer Datenbank legen wir eine Tabelle für jede Entity an, hingegen nie für eine Relationship. *Begründung: Wir erstellen eine eigene Tabelle für jede n-to-n Relationship.*		✓
Gegeben zwei Tabellen T_1 und T_2, die eine Grösse (Anzahl der Tupel) von s_1 bzw. s_2 haben, so ist die Grösse von $join(T_1, T_2)$ höchstens $s_1 \times s_2$. *Begründung: Je eine Zeile von T_1 und T_2 ergeben höchstens eine Zeile im join, egal welche Art von join wir verwenden.*	✓	
Zwei verschiedene Datenbanken haben immer unterschiedliche ER-Diagramme. *Begründung: Zwei verschiedene Datenbanken können laut Skript identische ER-Diagramme haben; beispielsweise könnte eine Datenbank nur eine Teilmenge der Datensätze (rows) der anderen Datenbank haben.*		✓
$B+$-trees können nur für Attribute verwendet werden, auf denen eine Ordnung definiert ist. *Begründung: $B+$-trees sind so gebaut, dass sie die verwalteten Elemente geordnet bereithalten; also müssen die Elemente anordenbar sein.*	✓	

2 Public Key Krypto (10 Punkte)

Gegeben sei eine öffentliche "collision resistant" Hashfunktion h, und ein sicheres Public Key Cryptosystem mit öffentlichem Schlüssel k_p von Alice, und einem geheimen Schlüssel k_s, den nur Alice kennt. Für eine Nachricht m gilt: $k_p(m) = c$ und $k_s(c) = m$.

a) Bob schickt $k_p(m), h(m)$ an Alice. Betrachten Sie zwei Fälle, bei denen eine Eavesdropperin Eve mithört. Kann Eve jeweils die Nachricht entschlüsseln?

(i) `[2 Punkte]` Eve erhält $k_p(m)$

(ii) `[2 Punkte]` Eve erhält $k_p(m)$ und $h(m)$

b) Bob schickt nun Nachrichten m_i, $i = 1, 2, 3 \ldots$, via $k_p(m_i), h(k_p(m_i))$ an Alice. Eve fängt die Nachrichten ab und kann sie verändert an Alice weiterschicken. Was gilt für die Vorgehensweise von Bob?

(i) `[2 Punkte]` Sie ist Malleable

(ii) `[2 Punkte]` Sie erfüllt Forward Secrecy

(iii) `[2 Punkte]` Sie ist sicher gegen Replay-Attacken

Lösungen

a) Bob schickt $k_p(m), h(m)$ an Alice. Betrachten Sie drei Fälle, bei denen eine Eavesdropperin Eve mithört. Kann Eve jeweils die Nachricht entschlüsseln?

(i) `[2 Punkte]` und (ii) `[2 Punkte]` Beides nein: Aus $k_p(m)$ erhalten wir keine Informationen (sicheres PKC), und h ist eine one-way Hashfunktion, da h collision resistant ist – erlaubt also auch keinen Rückschluss auf m.

b) Bob schickt nun Nachrichten m_i, $i = 1, 2, 3 \ldots$, via $k_p(m_i), h(k_p(m_i))$ an Alice. Eve fängt die Nachrichten ab und kann sie verändert an Alice weiterschicken. Was gilt für die Vorgehensweise von Bob?

(i) `[2 Punkte]` Ja, da der öffentliche Schluessel k_p und h öffentlich ist, kann die Nachricht verändert werden - Eve verschlüsselt und hashed ihre eigene Nachricht, welche sie dann an Alice schickt.

(ii) `[2 Punkte]` Nein, der geheime Schlüssel k_s ist für alle Nachrichten identisch.

(iii) `[2 Punkte]` Nein, Bob könnte auch zweimal die gleiche Nachricht schicken - dann kann Alice nicht unterscheiden ob Eve "replayed" oder ob Bob noch einmal sendet.

3 Hashing (16 Punkte)

a) [4] Gegeben seien eine Hashtabelle der Grösse m und ein Universum U mit $|U| > m$. Zeigen Sie, dass jede universelle Familie von Hashfunktionen mindestens m Funktionen enthält.
Hinweis: Eine Familie von Hashfunktionen $\mathcal{H}$ ist genau dann universell, wenn für jedes Paar von zwei verschiedenen keys k_1, k_2 gilt, dass bei uniformer Auswahl einer Hashfunktion $h \in \mathcal{H}$ gilt, dass $\Pr[h(k_1) = h(k_2)] = \frac{1}{m}$.

Wir betrachten jetzt eine konkrete Hashfunktion. Sei $U = \{0, \ldots, 2^w - 1\}^l$ für $w, l \geq 4$, l und w gerade, $l \leq 2^w$; das Universum besteht aus Tupeln der Länge l von Bitstrings der Länge w. Sei $m = 2^w$ die Grösse einer Hashtabelle. Für $k = (k_1, \ldots, k_l) \in U$ sei der Wert der Hashfunktion $h_\oplus : U \to \{0, \ldots, m-1\}$ an der Stelle k definiert als:

$$h_\oplus(k_1, \ldots, k_l) = k_1 \oplus \ldots \oplus k_l$$

wobei $\oplus$ bitweises XOR ist. Beispiel: $0011 \oplus 0101 = 0110$.

b) [6] Geben Sie explizit eine Menge von mindestens 2^l keys an, die unter $h_\oplus$ den selben Hashwert haben.

c) [6] Wenn n keys unabhängig voneinander uniformverteilt aus U ausgewählt werden, was ist die erwartete Anzahl an Kollisionen, die unter $h_{\oplus}$ entstehen?

Lösungen

a) Wenn $\mathcal{H}$ universell ist, dann gilt für jedes Paar von verschiedenen Schlüsseln k_1 und k_2, dass k_1 und k_2 unter genau $\frac{|\mathcal{H}|}{m}$ vielen Hashfunktionen in $\mathcal{H}$ kollidieren. Da die Anzahl von Funktionen, unter denen k_1 und k_2 kollidieren eine ganze Zahl sein muss, muss $\frac{|\mathcal{H}|}{m} \geq 1$ gelten. Wenn aber $|\mathcal{H}| < m$, dann gilt $\frac{|\mathcal{H}|}{m} < 1$. Also muss $|\mathcal{H}| \geq m$.

b) Sei $k = (k_1, \ldots, k_l)$ die Binärdarstellung von $(0, 1, 2, \ldots, l-1)$. Wir können nun k beliebig permutieren, um einen neuen, anderen key zu erhalten, der unter $h_\oplus$ den selben Hash wie k hat. Wir erhalten so $l!$ viele keys mit dem selben Hash, und weil $l \geq 4$ gilt $l! \geq 2^l$.

c) Jeweils $\frac{|U|}{2^w} = \frac{|U|}{m}$ viele keys haben unter $h_\oplus$ den selben Hash, Beweis: Nimm für einen beliebigen key $k = (k_1, \ldots, k_l)$ die XOR-Summe der ersten $l-1$ Komponenten, dann entscheidet k_l die Gesamtsumme. Weil es 2^w Möglichkeiten für k_l gibt und jede davon eine andere Gesamtsumme erzeugt, haben alle diese 2^w keys einen anderen Hash.

Wenn wir zwei keys uniformverteilt aus U ziehen, haben wir also eine Chance von $\frac{1}{m}$ zwei keys zu ziehen, die unter $h_\oplus$ kollidieren. Mit Linearität der Erwartung ergibt sich damit $\mathbb{E}[\mathit{Anzahl\ Kollisionen}] = \binom{n}{2}\frac{1}{m}$.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2016 Prof. R. Wattenhofer

Technische Informatik II

Klausur

Samstag, 11. Februar 2017, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten sofern nichts anderes dabeisteht. Beschriften Sie Skizzen und Zeichnungen verständlich. Was wir nicht lesen können, können wir auch nicht bewerten!

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie **jedes Zusatzblatt** ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Name	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - Verschlüsseltes Testament		12
2 - SQL		23
Summe		**35**

1 Verschlüsseltes Testament (12 Punkte)

Bob will sein Testament T (encodiert als Bitstring) verschlüsselt hinterlassen. Er hinterlässt drei Personen drei verschiedene geheime Nachrichten N_1, N_2, N_3: seinem Notar, seiner Tochter, und seinem Geschäftspartner. Jede der drei Personen erfährt nur die ihr hinterlassene geheime Nachricht N_i. Der Einfachheit halber will Bob zum Ver- und Entschlüsseln nur die Operationen $+, -$ sowie bitweises AND, OR, NOT verwenden.

a) `[5 Punkte]` Bob will, dass jemand das Testament nur entschlüsseln kann, wenn er alle drei geheimen Nachrichten N_1, N_2, N_3 hat, und dass ohne alle drei zusammen Perfect Secrecy gewährleistet ist. Wie kann Bob die drei geheimen Nachrichten konstruieren, sodass diese Anforderungen gewährleistet sind?

b) `[7 Punkte]` Bob will zur Ausfallsicherheit das System verändern: das Testament soll nun genau dann entschlüsselbar sein, wenn man mindestens zwei beliebige geheime Nachrichten hat (egal welche zwei), und ansonsten soll Perfect Secrecy gewährleistet sein. Wie kann Bob die drei geheimen Nachrichten konstruieren, um das zu erreichen?

Lösungen

a) [5 Punkte] Bob wählt unabhängig voneinander uniform zwei Schlüssel k_1 und k_2 derselben Länge wie T aus und setzt $N_1 = k_1$, $N_2 = k_2$ und $N_3 = T \oplus k_1 \oplus k_2$. Es handelt sich um $(3, 3)$-Threshold Secret Sharing, das laut Vorlesung perfect secrecy gewährleistet, wenn nicht alle Teilnehmer kollaborieren.

b) [7 Punkte] Nun wählt Bob unabhängig voneinander uniform drei Schlüssel k_1, k_2, k_3 derselben Länge wie T aus und setzt die Nachrichten auf $N_1 = (T \oplus k_1, k_2, 1)$ und $N_2 = (T \oplus k_2, k_3, 2)$ und $N_3 = (T \oplus k_3, k_1, 3)$. Für zwei beliebige verschiedene Nachrichten $N_i = (C_i, k_i, i) \neq (C_j, k_j, j) = N_j$ gilt entweder $C_i \oplus k_j = T$ oder $C_j \oplus k_i = T$, und anhand der Indizes i und j ist klar, welches der beiden Ergebnisse stimmt. Lediglich eine Nachricht genügt zum Entschlüsseln nicht, da es sich dann um eine OTP-Verschlüsselung handelt, die perfectly secret ist.

2 SQL (23 Punkte)

In dieser Aufgabe verwenden wir die Film-Datenbank aus der Vorlesung und der Übung. Das Schema ist in Abbildung 1 dargestellt. Das Schema enthält nur die Unique Identifier und Foreign Keys für jede Tabelle. Die Namen einiger für die Aufgabe relevanter Spalten sind in Tabelle 1 aufgelistet.

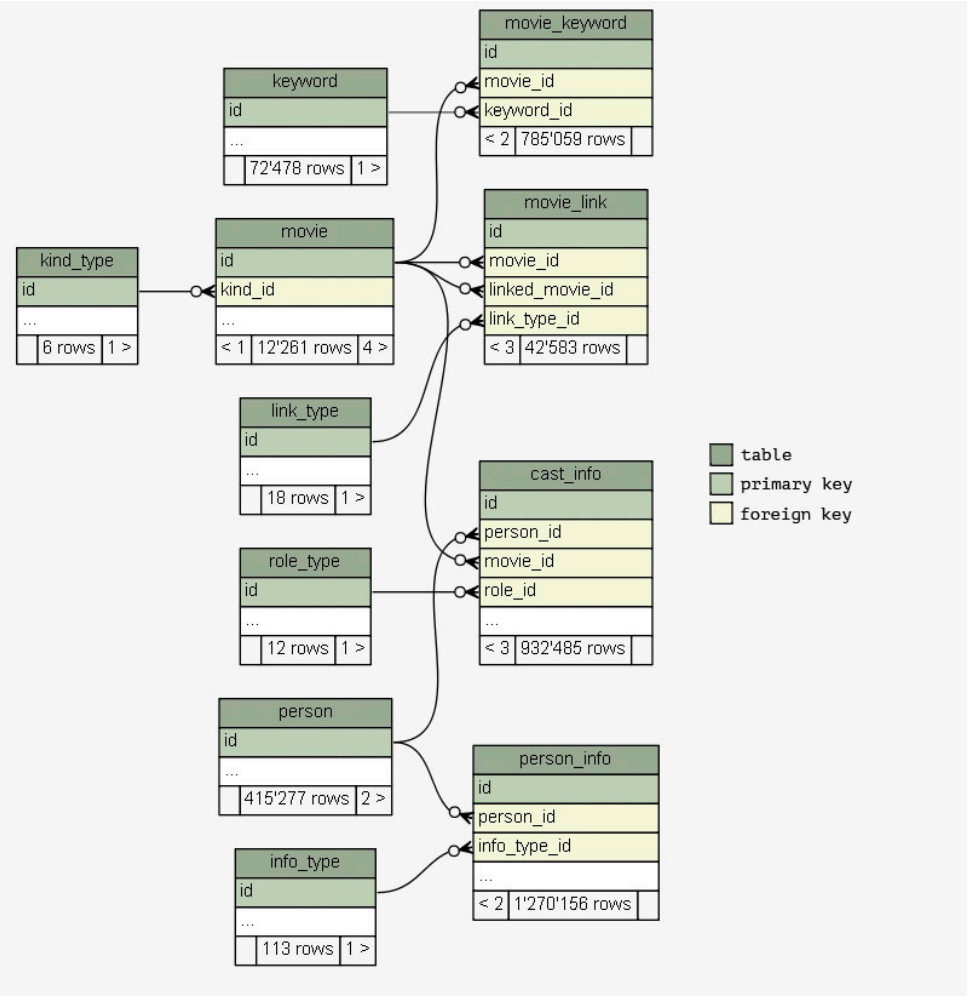


Abbildung 1: Datenbankschema welches die Tabellen mit ihren Primary und Foreign Keys zeigt.

movie
id
kind_id
title
year
tomatometer

person
id
name
gender

Tabelle 1: Relevante Spaltennamen der Datenbanktabellen *movie* und *person*.

a) Geben Sie für jeden der nachfolgenden Punkte an, welche Art von JOIN man verwenden müsste um eine korrekte und möglichst kompakte MySQL-Query zu schreiben. Sie müssen KEINE MySQL-Query schreiben.

(i) [2] Finden Sie für jeden **movie** die zugehörigen **movie_keyword**s.

(ii) [2] Geben Sie alle **person**en aus welche eine zugehörige **person_info** haben.

(iii) [2] Finden Sie heraus, wie viele **person**en kein zugehöriges **person_info** haben.

b) [4] Schreiben Sie eine MySQL-Query um die Anzahl **movie**s welche das Wort "of" im *title* enthalten auszugeben.

c) [6] Schreiben Sie eine MySQL-Query, die ausgibt wie viele männliche ("m") **person** mit Vornamen "Robert" beim **movie** "The Departed" mitgespielt haben. Sie können annehmen, dass kein Schauspieler mehr als eine Rolle spielt.

d) [4] Wir suchen eine MySQL-Query welche für jeden *tomatometer* grösser als 74 die Anzahl **movie**s mit dieser Bewertung finden soll. Das Resultat sollte absteigend nach der Anzahl gefundener **movie**s sortiert sein. Das Resultat der Query sollte also 25 Zeilen haben. Ein Freund schlägt folgende Query vor:

```
SELECT tomatometer,COUNT(title) AS cnt
FROM movie
WHERE tomatometer>74
ORDER BY cnt DESC;
```

Wenn Sie die Query laufen lassen, hat das Resultat nur eine Zeile. Wo liegt der Fehler, und wie können Sie die Query reparieren?

e) [3] MySQL kennt keine FULL OUTER JOINS. Wie könnte man mit bestehenden Befehlen dennoch einen FULL OUTER JOIN ausführen?

Lösungen

a) Joins

(i) Left Join

(ii) Join or Right Join

(iii) Left Join

```
SELECT COUNT(title)
FROM movie
WHERE title LIKE "% of %";
```

b)

```
SELECT COUNT(person.name)
FROM cast_info
JOIN movie ON movie.id = cast_info.movie_id
JOIN person ON person.id = cast_info.person_id
WHERE movie.title='The Departed' and person.gender='m'
AND person.name LIKE '%Robert%';
```

c)

d) Eve hat den "GROUP BY tomatometer" Befehl vergessen. ODER: GROUP BY und HAVING

```
SELECT tomatometer,COUNT(title) AS cnt
FROM movie
WHERE tomatometer>74
GROUP BY tomatometer
ORDER BY cnt DESC;
```

e) Die Resultate von Left und Right Outer Join kombinieren. (Eigentlich mit UNION, aber es ist nicht nötig dies explizit zu schreiben)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2017 Prof. R. Wattenhofer

Technische Informatik II

Klausur Mittwoch, 23. August 2017, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten sofern nichts anderes dabeisteht. Beschriften Sie Skizzen und Zeichnungen verständlich. Was wir nicht lesen können, können wir auch nicht bewerten!

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie **jedes Zusatzblatt** ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Familienname	**Vorname**	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - Multiple Choice		9
Summe		**9**

1 Multiple Choice (9 Punkte)

Geben Sie bei jeder Aussage an, ob sie wahr oder falsch ist. Jede korrekte Antwort gibt 1 Punkt, **jede fehlerhafte Antwort -1 Punkt**. Jede unbeantwortete Aussage gibt 0 Punkte. Wenn die Summe der Punkte für alle Aussagen negativ ist, dann wird die Aufgabe insgesamt mit 0 Punkten bewertet. **In dieser Aufgabe können Sie Ihre Antworten nicht begründen.**

Aussage	**wahr**	**falsch**
HAVING und WHERE Klauseln können als Synonyme füreinander verwendet werden.	☐	☐
Wenn wir JOINs benutzen benötigen wir sogenannte JOIN Tabellen.	☐	☐
Ein Foreign Key kann nur einen Primary Key referenzieren.	☐	☐
Jede Hashfunktion $h : U \to M$ erzeugt Kollisionen auf jeder Schlüsselmenge $N \subseteq U$.	☐	☐
Perfect Static Hashing garantiert, dass jede Suche konstant viel Zeit kostet.	☐	☐
Wenn Hashes uniform verteilt für eine Hashtabelle der Grösse $m = 1000$ gewählt werden, dann ist die **Wahrscheinlichkeit einer Kollision** bei **100** Schlüsseln bereits 50%.	☐	☐
Wenn Hashes uniform verteilt für eine Hashtabelle der Grösse $m = 1000$ gewählt werden, dann ist die **Wahrscheinlichkeit einer Kollision** bei **200** Schlüsseln bereits 50%.	☐	☐
Wenn Hashes uniform verteilt für eine Hashtabelle der Grösse $m = 1000$ gewählt werden, dann ist bei **500** Schlüsseln **eine Kollision garantiert**.	☐	☐
Die Sicherheit vieler Public-Key-Verschlüsselungsverfahren beruht darauf, dass es schwer ist grosse Primzahlen zu finden.	☐	☐

Lösungen

Aussage	**wahr**	**falsch**
HAVING und WHERE Klauseln können als Synonyme füreinander verwendet werden. *Begründung: Nach GROUP BY müssen wir HAVING benutzen.*		✓
Wenn wir JOINs benutzen benötigen wir sogenannte JOIN Tabellen. *Begründung: Man kann beliebige Tabellen Joinen. Join-Tabellen sind einfach dazu gedacht die Beziehungen zwischen Tabellen explizit darzustellen.*		✓
Ein Foreign Key kann nur einen Primary Key referenzieren. *Begründung: Es muss ja ein 1-1 mapping sein. Ein Key kann nicht mehrere Primary Keys referenzieren, da das mapping gar nicht definiert wäre.*	✓	
Jede Hashfunktion $h : U \to M$ erzeugt Kollisionen auf jeder Schlüsselmenge $N \subseteq U$. *Begründung: Wähle $\|M\| \geq \|N\|$, und lasse h so sein, dass h jeden Schlüssel $k \in N$ auf einen eigenen Bucket abbildet. Oder trivialer Fall: sei $\|N\| \leq 1$, dann gibt es keine zwei Schlüssel, die kollidieren könnten.*		✓
Perfect Static Hashing garantiert, dass jede Suche konstant viel Zeit kostet. *Begründung: In Perfect Static Hashing kostet nur die Konstruktion der Tabelle in Erwartung linear viel Zeit, Suche kostet immer (also auch worst-case) konstant viel.*	✓	
Wenn Hashes uniform verteilt für eine Hashtabelle der Grösse $m = 1000$ gewählt werden, dann ist die **Wahrscheinlichkeit einer Kollision** bei **100** Schlüsseln bereits 50%. *Begründung: Siehe Birthday Problem Rechnung im Skript.*	✓	
Wenn Hashes uniform verteilt für eine Hashtabelle der Grösse $m = 1000$ gewählt werden, dann ist die **Wahrscheinlichkeit einer Kollision** bei **200** Schlüsseln bereits 50%. *Begründung: Siehe Birthday Problem Rechnung im Skript.*	✓	
Wenn Hashes uniform verteilt für eine Hashtabelle der Grösse $m = 1000$ gewählt werden, dann ist bei **500** Schlüsseln **eine Kollision garantiert**. *Begründung: Dafür benötigt es mindestens* 1001 *Schlüssel, da sonst per Zufall alle Schlüssel in einen eigenen Bucket kommen könnten.*		✓
Die Sicherheit vieler Public-Key-Verschlüsselungsverfahren beruht darauf, dass es schwer ist grosse Primzahlen zu finden. *Begründung: Grosse Primzahlen finden ist einfach.*		✓

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2017 Prof. R. Wattenhofer

Technische Informatik II

Klausur

Mittwoch, 14. Februar 2018, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten sofern nichts anderes dabeisteht. Beschriften Sie Skizzen und Zeichnungen verständlich. Was wir nicht lesen können gibt keine Punkte!

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie **jedes Zusatzblatt** ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Familienname	**Vorname**	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - Projektplanung		25
2 - Security		20
Summe		**45**

1 Projektplanung (25 Punkte)

An der EPFL müssen die Studierenden $S = \{S_1, \ldots, S_n\}$ jedes Semester Projekte $P = \{P_1, \ldots, P_m\}$ wählen. Dazu geben sie zu Beginn des Semesters an, welche Projekte sie interessieren. Anschliessend weist das System den Studierenden aus ihrer Auswahl Projekte zu.

a) `[10 Punkte]` Zunächst darf ein Projekt von maximal einer Studierenden belegt werden und Studierende wollen maximal ein Projekt belegen. Wir wollen das Problem, eine maximale Anzahl solcher 1:1-Zuordnungen zu finden, als Maximalflussproblem modellieren, siehe Abbildung 1. Eine Kante (S_i, P_j) zeigt dabei an, dass Studierende S_i an Projekt P_j interessiert ist. Wie müssen die Kapazitäten der drei Arten von Kanten $((s, S_i), (S_i, P_j), (P_j, t))$ gewählt werden, damit ein *ganzzahliger* Maximalfluss einer zulässigen Zuordnung entspricht?
Geben Sie auch eine zulässige Zuordnung zwischen Studierenden $S = \{S_1, S_2, S_3, S_4, S_5\}$ und Projekten $P = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ an, sodass alle Studierenden in Abbildung 1 genau ein Projekt erhalten.

Hinweis: Sind die Kapazitäten ganzzahlig, existiert auch immer ein ganzzahliger Fluss.

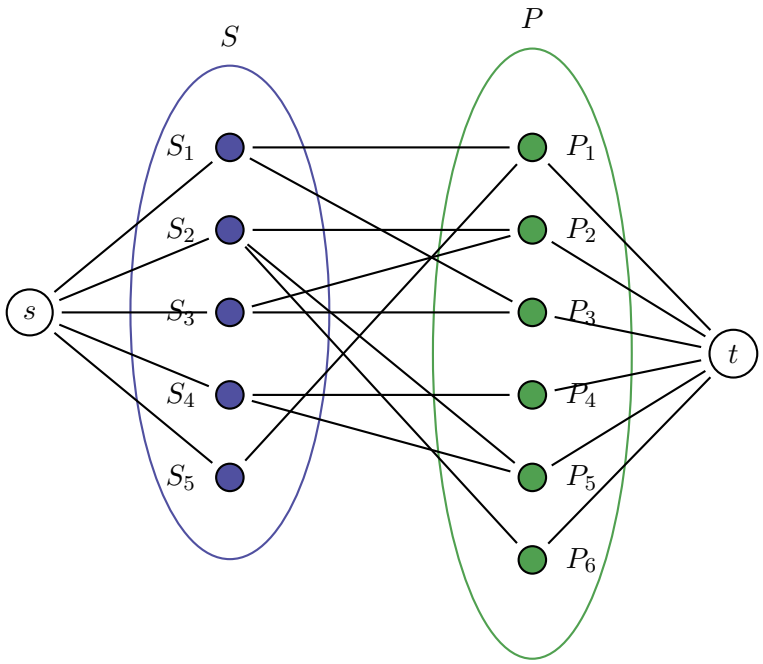


Abbildung 1: Netzwerk zu Aufgabe a)

b) [3 Punkte] Welchen Wert hat der Fluss im allgemeinen Fall, wenn jede(r) Studierende genau ein Projekt erhält?

c) [5 Punkte] Ein Projekt P_j kann jetzt von einer vorher definierten Anzahl Studierender $n_j \geq 1$ belegt werden. Wie muss der Graph modifiziert werden, um die Projektplanung auch mit dieser Erweiterung als Maximalflussproblem zu modellieren?

d) [7 Punkte] In einer weiteren Modifikation darf jeder Student nun eine Höchstanzahl an Projekten angeben. Wie muss das Netzwerk für das Maximalflussproblem dafür angepasst werden? Ein Professor schlägt für dieses Problem einen einfachen Algorithmus vor: Weise den Studierenden reihum solange Projekte zu, die sie interessieren, bis keine Zuweisungen mehr möglich sind.
Zeigen Sie an einem Beispiel, dass mit dieser Vorgehensweise nicht unbedingt die maximale Anzahl an Zuweisungen erreicht wird.

Lösungen

a) Eine zulässige Lösung ist $(S_1, P_3), (S_2, P_5), (S_3, P_2), (S_4, P_4), (S_5, P_1)$. Die Kapazitäten setzen wir auf 1 (für die Kanten (S_i, P_j) in der Mitte ist die Kapazität egal). Dann entspricht ein maximaler integraler Fluss einem maximum Matching und damit einer zulässigen Zuordnung.

b) Der Wert des Flusses entspricht der Anzahl der Studierenden, wenn jedem genau ein Projekt zugewiesen wurde.

c) Die Kanten (P_l, t) für $l = 1, \ldots, m$ bekommen Anzahl der verfügbaren Plätze für Kurs P_l, die Kanten (S_i, P_j) müssen jetzt auch Kapazität 1 haben.

d) Jetzt gibt es auch Kapazitäten auf den Kanten (s, S_i) entsprechend der Höchstzahl an Projekten pro Student. Wenn im folgenden Beispiel zunächst S_1 (1 Kurs) zu P_2 und dann S_2 (2 Kurse) zu P_3 zugewiesen wurde, ist keine weitere Zuweisung mehr möglich. Allerdings wäre die Zuweisung $S_1 \mapsto P_1, S_2 \mapsto P_2, S_2 \mapsto P_3$ optimal.

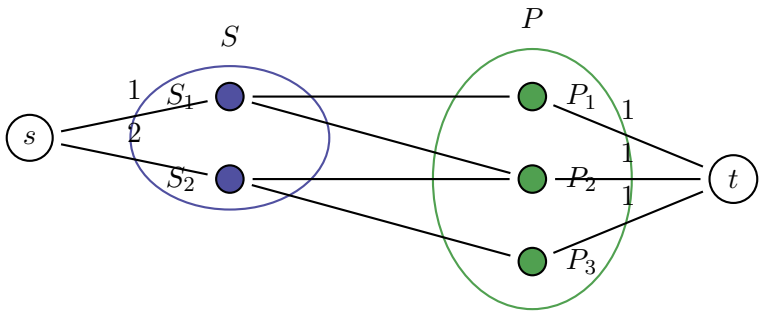


Abbildung 2: Beispiellösung d)

2 Security (20 Punkte)

Wir betrachten verschiedene Szenarien in denen A(lice) und B(ob) miteinander kommunizieren möchten, und E(ve) einen Angriff ausführt. Geben Sie für jedes Szenario an ob es realistisch ist und begründen Sie.

a) [3] A und B haben einen sicheren Kommunikationskanal mit geteiltem Schlüssel. Der Schlüssel ist sicher, nur A und B kennen ihn. E möchte wissen, was A und B besprechen. Dazu gibt sich E gegenüber A als B aus und gegenüber B als A. Da alle Kommunikation jetzt über E läuft, ist E Man-in-the-middle und kann den Inhalt der Nachrichten lesen.

b) [3] A möchte Geld auf B's Bankkonto überweisen. E hat ein Gerät zum überwachen und weiterleiten des Netzwerkverkehrs an A's Router angebracht. Wenn A die Zahlung in Auftrag gibt, fängt E die Nachricht ab, ändert den Zahlungsempfänger von B zu sich selbst, und leitet die Zahlung dann weiter an A's Bank.

c) [3] A und B möchten einen gemeinsamen geheimen Schlüssel generieren mit dem Diffie-Hellman Algorithmus 13.14 aus dem Skript. E kann die Schwächen des Algorithmus ausnutzen um man-in-the-middle zu werden.

d) [3] A und B haben je ein sicheres public-private key pair welches sie benutzen um Nachrichten zu signieren. Sie möchten nun einen gemeinsamen geheimen Schlüssel erstellen mit dem DH Algorithmus. E hört seit längerer Zeit der Kommunikation zwischen A und B zu, und hat ab und zu Nachrichten erneut gesendet (replay). E schafft es jetzt während dem DH Austausch zwischen A und B sich als Man-in-the-middle zu etablieren.

e) [3] A und B verwenden One-Time Pads für sichere Kommunikation. E kennt A und B im echten Leben und hört wie sie von einem wichtigen Geheimnis sprechen, welche A vor einiger Zeit an B gesendet hat. Eines Abends ist B betrunken, und E schafft es einige von B's geheimen Schlüsseln zu stehlen. Da E schon seit langer Zeit die verschlüsselte Kommunikation zwischen A und B aufgezeichnet hat, kann E jetzt endlich das wichtige Geheimnis entschlüsseln.

f) [5] A und B möchten ein eigenes Kommunikationsprotokoll entwerfen. A hat folgenden Vorschlag: Bei jeder Kommunikationrunde wird ein zufälliger Initialisierungsvektor c_0 der Länge $l > 0$ **sicher** ausgetauscht. Der cipher text c_i von Klartextblock m_i (ebenfalls jeweils der Länge l) wird berechnet als $c_i := i \oplus (m_i \oplus c_{i-1})$ für $i = \{1, 2, \ldots\}$. Eine neue Kommunikationsrunde fängt spätestens dann an, wenn $i = 2^l - 1$. Der Index i wird als Binärzahl der Länge l dargestellt. E kann den Inhalt aller Nachrichten herausfinden, indem E nur die verschlüsselten Nachrichten c_i mitliest.

Lösungen

a) (i) Nicht realistisch. Eve kann nicht MITM werden weil schon ein sicherer Kommunikationschannel besteht. Eve kann sich also nicht als A oder B ausgeben.

(ii) Nicht realistisch. Bankenverkehr ist verschlüsselt, Eve müsste bereits MITM sein. Es reicht nicht, einfach den traffic passiv zu überwachen. Zudem kann man Replay Attacken ausführen ohne MITM zu sein.

(iii) Realistisch. Algorithmus 13.14 ist anfällig gegen MITM, da sich Eve während dem DH Protokoll gegenüber A als B und gegenüber B als A ausgeben kann. A und B denken fälschlicherweise, dass sie direkt miteinander kommunizieren.

(iv) Nicht realistisch. Eve kann die Signaturen nicht fälschen und kann deshalb nicht MITM sein, da sie sich nicht als A und B ausgeben kann.

(v) Realistisch (je nachdem aber eher unwahrscheinlich). Eve muss einfach hoffen, dass sich das richtige one-time pad per Zufall unter den gestohlenen befindet.

(vi) Nicht realistisch. m_1 kann nicht herasugefunden werden, da Eve c_0 nicht kennt. Alle nachfolgenden Nachrichten kann Eve jedoch entschlüsseln. Betrachten wir $c_{i+1} = (i+1)(m_{i+1} \oplus c_i)$. Wenn wir c_i auch kennen, dann haben wir

$$\begin{aligned} c_{i+1} &= (i+1)(m_{i+1} \oplus c_i) \\ c_{i+1}(i+1)^{-1} &= m_{i+1} \oplus c_i \\ c_{i+1}(i+1)^{-1} \oplus c_i &= m_{i+1} \end{aligned}$$

Wir können also m_{i+1} direkt ausrechnen, wenn wir c_{i+1} und c_i kennen.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2018 Prof. R. Wattenhofer

Technische Informatik II

Klausur

Mittwoch, 8. August 2018, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten sofern nichts anderes dabeisteht. Beschriften Sie Skizzen und Zeichnungen verständlich. Was wir nicht lesen können gibt keine Punkte!

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie **jedes Zusatzblatt** ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Familienname	**Vorname**	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - Multiple Choice		3
2 - Flow Control mit LPs		20
3 - Hashing		20
Summe		**43**

1 Multiple Choice (3 Punkte)

Geben Sie bei jeder Aussage an, ob sie wahr oder falsch ist. Jede korrekte Antwort gibt 1 Punkt. Jede fehlerhafte Antwort und unbeantwortete Aussage gibt 0 Punkte. **In dieser Aufgabe können Sie Ihre Antworten nicht begründen oder erklären.**

a) `[3 Punkte]` Wir nehmen an, ein Angreifer erfährt den Klartext und den dazugehörigen Ciphertext einer Nachricht. Der Ciphertext besteht aus Cipher Block Chaining (CBC) Blöcken.

	wahr	**falsch**
Wenn die Nachricht durch das Shiften jedes Zeichens um k Positionen im Alphabet verschlüsselt wurde kann der Angreifer den Schlüssel k herausfinden.	☐	☐
Wenn jeder Block durch ein anderes One-Time-Pad verschlüsselt ist, kann der Angreifer jedes der One-Time-Pads bestimmen.	☐	☐
Durch die Verwendung des CBC-Modus kann es für den Angreifer schwieriger sein, den Schlüssel zu finden als beim ECB-Modus.	☐	☐

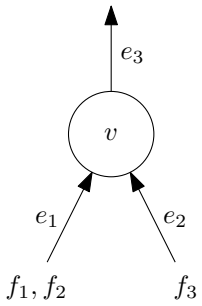
Lösungen

a) [3 Punkte] Wir nehmen an, ein Angreifer erfährt den Klartext und den dazugehörigen Ciphertext einer Nachricht. Der Ciphertext besteht aus Cipher Block Chaining (CBC) Blöcken.

	wahr	**falsch**
Wenn die Nachricht durch das Shiften jedes Zeichens um k Positionen im Alphabet verschlüsselt wurde kann der Angreifer den Schlüssel k herausfinden. *Begründung: k kann bestimmt werden durch das Subtrahieren des ersten Zeichens des Klartextes vom ersten Zeichen des Ciphertextes.*	✓	
Wenn jeder Block durch ein anderes One-Time-Pad verschlüsselt ist, kann der Angreifer jedes der One-Time-Pads bestimmen. *Begründung: Jedes One-Time-Pad kann durch XOR des verschlüsselten Blocks mit dem Klartext-Block und dem vorherigen Ciphertext-Block bestimmt werden.*	✓	
Durch die Verwendung des CBC-Modus kann es für den Angreifer schwieriger sein, den Schlüssel zu finden als beim ECB-Modus. *Begründung: Der Angreifer kennt den Klartext und den Ciphertext der Nachricht, der CBC-Modus hat dadurch keinen Einfluss auf die Schwierigkeit des Bestimmens des Schlüssels.*		✓

2 Flow Control mit LPs (20 Punkte)

Flow-Probleme können nicht immer global gelöst werden. In manchen Fällen (z.B. in einem Router) muss lokal entschieden werden, welche Flows mit welcher Geschwindigkeit weitergeleitet werden. Im Folgenden betrachten wir ein *lokales* Flow-Problem im Knoten v, das heisst, gegeben die eingehenden Flows f_i (mit $1 \leq i \leq 3$) auf den Kanten e_1 und e_2 müssen wir entscheiden, welche Flows mit den Raten r_i auf der Kante e_3 weitergeleitet werden. Die Kanten haben Kapazitäten $c(e_1) = c(e_2) = 3$ und $c(e_3) = 5$. Jeder Flow f_i ist splittable und hat Demand $d_i = i$, also $d_1 = 1$, $d_2 = 2$ und $d_3 = 3$. Falls mehrere Flows eine Kante passieren, darf die Summe ihrer Raten die Kapazität der Kante nicht übersteigen.



Es gelten ausserdem folgende Prioritäten: Flow f_1 ist **sehr** wichtig, Flow f_2 ist wichtig und Flow f_3 ist weniger wichtig.

a) `[4 Punkte]` Stellen Sie ein LP auf, das eine Allokation der gegebenen Flows unter Berücksichtigung der Prioritäten berechnet.

b) `[6 Punkte]` Eine Studentin behauptet, sie könne einfach die Flows f_1 und f_2 zusammenfassen zu f_x mit der Rate $r_x = r_1 + r_2$. Stellen Sie das resultierende LP für f_3 und f_x auf, und lösen Sie dieses mit dem Simplex-Algorithmus beginnend bei $(0, 0)$. Wählen Sie dazu eine geeignete, möglichst einfache Funktion, die vom Simplex-Algorithmus maximiert werden soll und die Prioritäten abbildet.

c) `[6 Punkte]` Angenommen der Flow f_3 kann nur vollständig ($r_3 = d_3$) oder gar nicht ($r_3 = 0$) zugewiesen werden. Weiterhin muss die Kapazität der ausgehenden Kante e_3 vollständig ausgenutzt werden. Könnte dieses Problem dennoch mit einem LP gelöst werden?

d) `[4 Punkte]` Geben Sie für beide Fälle (Flow f_3 ist teilweise zuweisbar/nur vollständig zuweisbar) eine Max-Min-Fair Allokation der Flows an, wenn die ausgehende Kante e_3 jeweils vollständig ausgelastet sein muss.

Lösungen

a)

$$\begin{array}{lll} r_1 \geq 0 & r_1 \leq 1 = d_1 & r_1 + r_2 \leq 3 = c(e_1) \\ r_2 \geq 0 & r_2 \leq 2 = d_2 & r_1 + r_2 + r_3 \leq 5 = c(e_3) \\ r_3 \geq 0 & r_3 \leq 3 = d_3 & \end{array}$$

Nun muss eine Funktion maximiert werden, die die Prioritäten adäquat umsetzt. Zum Beispiel:

$$f_1(r) = \underbrace{3r_1}_{\textbf{sehr } wichtig} + \underbrace{2r_2}_{wichtig} + \underbrace{r_3}_{weniger\ wichtig} .$$

b)

$$r_x = r_1 + r_2 \qquad \Longrightarrow\ r_x \leq 3 = d_x = d_1 + d_2 = c(e_1)$$

Das neue LP ist also:

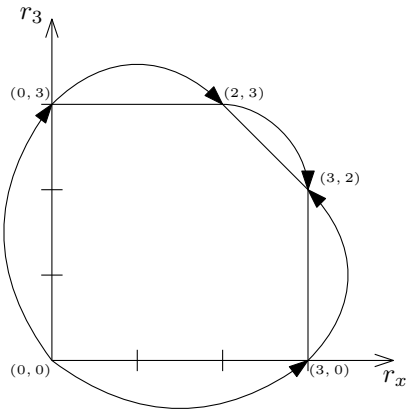
$$\begin{array}{lll} r_x \geq 0 & r_x \leq 3 = d_x & r_x + r_3 \leq 5 = c(e_3) \\ r_3 \geq 0 & r_3 \leq 3 = d_3 & \end{array}$$

Wir maximieren nun die Funktion:

$$f_2(r) = \underbrace{2r_x}_{\textbf{sehr } wichtig\,/\,wichtig} + \underbrace{r_3}_{weniger\ wichtig} .$$

Daraus ergeben sich zwei korrekte Lösungen mit dem Simplex-Algorithmus:

- $(0,0) \to (3,0) \to (3,2)$
- $(0,0) \to (0,3) \to (2,3) \to (3,2)$



c) Man kann hier beobachten, dass $d_1 + d_2 = 3$ noch zu wenig sind, um e_3 ganz auszulasten. Deshalb muss $r_3 = d_3 = 3$ gelten und r_1 bzw. r_2 können durch folgendes LP bestimmt werden:

$$\begin{array}{lll} r_1 \geq 0 & r_1 \leq 1 = d_1 & r_1 + r_2 \leq 2 = c(e_3) - r_3 \\ r_2 \geq 0 & r_2 \leq 2 = d_2 & \end{array}$$

Wir maximieren nun die Funktion:

$$f_3(r) = \underbrace{2r_1}_{\textbf{sehr } wichtig} + \underbrace{r_2}_{wichtig} .$$

d)

- f_3 **teilweise zuweisbar:** $r_1 = 1,\ r_2 = 2,\ r_3 = 2$
- f_3 **nur vollständig zuweisbar:** $r_1 = 1,\ r_2 = 1,\ r_3 = 3$

3 Hashing (20 Punkte)

Zunächst betrachten wir das sogenannte Zufallshashing: Gegeben ist eine zufällige Permutation der Zahlen von 1 bis m. Die Probing-Sequence ist

$$h_i(k) = (h(k) + r_i) \ mod \ m,$$

wobei r_i das i-te Element der Permutation ist.

a) `[4 Punkte]` Leidet Zufallshashing an primärem Clustering?

b) `[4 Punkte]` Leidet Zufallshashing an sekundärem Clustering?

Ein Ingenieur schlägt vor, ein dynamisches Dictionary mit einer rekursiven Probing-Sequence zu implementieren. Konkret schlägt er vor, ein Objekt k in die Zelle $h(k)$ einzufügen, falls diese leer ist, sonst in $h(h(k))$, falls diese leer ist, sonst in $h(h(h(k)))$, falls diese leer ist, etc. Somit wäre die rekursive Probing-Sequence:

$$h_1(k) = h(k)$$
$$h_{i+1}(k) = h(h_i(k)).$$

c) `[6 Punkte]` Wie kann das Löschen in so einem rekursiven Dictionary funktionieren?

d) `[6 Punkte]` Welches gravierende Problem kann beim Einfügen auftreten?

Lösungen

a) Nein. Wenn zwei Probing-Sequences in der gleichen Zelle landen, werden diese nicht erneut in der darauffolgenden Zelle kollidieren da der jeweilige additive Offset unterschiedlich ist bei unterschiedlichem i. Es kommt also nicht zu primärem Clustering.

b) Ja. Wenn zwei Objekte den gleichen Hash-Wert besitzen, werden diese beiden fortwährend der gleichen Probing-Sequence unterliegen da bei beiden die selben r_i addiert werden. Es kommt also zu sekundärem Clustering.

c) Trifft man bei einer Löschoperation auf eine leere Zelle in der rekursiven Probing-Sequence, so ist unklar ob dort ein Wert gelöscht wurde, oder aber niemals ein Wert in dieser Zelle vorhanden war. Dieses Problem tritt auch bei anderen Probing-Sequences auf. Eine Möglichkeit dies dennoch umzusetzen wäre, Felder mit einem „gelöscht“- bzw. „continue“-Flag zu markieren.

d) Beim Einfügen ergibt sich das Problem, dass die rekursive Probing-Sequence möglicherweise nach kurzer Zeit zu einem bereits besuchten Feld zurückkehrt und so ein neuer Wert trotz Kapazität in der Tabelle nicht eingefügt werden kann. Insbesondere sei hier auf die gängige Hashfunktion $h(k) = k \ mod \ p$ hingewiesen, es müsste also bei der Auswahl der Hashfunktion auf diese Eigenschaft geachtet werden.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2018 Prof. R. Wattenhofer

Technische Informatik II

Klausur

Mittwoch, 23. Januar 2019, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. **Begründen Sie alle Ihre Antworten** sofern nichts anderes steht. Beschriften Sie Skizzen und Zeichnungen verständlich. **Was wir nicht lesen können gibt keine Punkte!**

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie jedes Zusatzblatt ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Familienname	**Vorname**	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - Transport Layer: LPs		15
Summe		**15**

1 Transport Layer: LPs (15 Punkte)

a) `[7 Punkte]` Wir betrachten das folgende LP:

Maximiere: $f(x) = 7x_1 - 12x_2$ unter den Bedingungen

1: $x_1 \geq 0$
2: $x_1 \leq 4$
3: $x_2 \geq 0$
4: $x_2 \leq 3$
5: $x_1 - x_2 \leq 3$

Führen Sie den Simplex Algorithmus beginnend bei $(0, 0)$ durch und beantworten Sie folgende Fragen:

(i) Was sind die Eckpunkte des Polytops?

(ii) Was ist die Lösung des LPs?

Hinweis: *Eine aussagekräftig beschriftete Skizze ist als Lösung zugelassen.*

b) `[8 Punkte]` Wir betrachten das folgende LP:

Maximiere: $f(x) = 3x_1 - x_2 + x_3$ unter den Bedingungen

1: $x_1 \geq 1$
2: $-x_1 \geq -5$
3: $2x_2 \geq 0$
4: $\frac{1}{3}x_2 \leq 1$
5: $x_3 \geq 1$
6: $2x_3 \leq 2$
7: $x_1 - 1 \geq 2x_2$
8: $5 - x_1 + x_2 - 2x_3 \geq 0$

Berechnen Sie die Lösung des LPs.

Hinweis: *Der Punkt* $(1, 0, 1)$ *ist ein Eckpunkt des Polytops.*

Lösungen

a) Die Eckpunkte des Polytops sind:

- $x = (0, 0)$,
- $x = (3, 0)$,
- $x = (4, 1)$,
- $x = (4, 3)$,
- $x = (0, 3)$.

Der Simplex-Algorithmus findet das Ergebnis in einem Schritt: $(0, 0) \to (3, 0)$.

b) Zunächst beobachten wir, dass:

- $-x_1 \geq -5$ äquivalent zu $x_1 \leq 5$ ist,
- $2x_2 \geq 0$ äquivalent zu $x_2 \geq 0$ ist,
- $\frac{1}{3}x_2 \leq 1$ äquivalent zu $x_2 \leq 3$ ist,
- $x_3 \geq 1$ und $2x_3 \leq 2$ impliziert, dass $x_3 = 1$ gilt,
- $x_1 - 1 \geq 2x_2$ äquivalent zu $x_1 - 2x_2 \geq 1$ ist,
- und $5 - x_1 + x_2 - 2x_3 \geq 0$ nun äquivalent zu $x_1 - x_2 \leq 3$ ist.

Wir erhalten das LP:
Maximiere: $f(x) = 3x_1 - x_2 + 1$ unter den Bedingungen
1: $x_1 \geq 1$
2: $x_1 \leq 5$
3: $x_2 \geq 0$
4: $x_2 \leq 3$
5: $x_1 - 2x_2 \geq 1$
6: $x_1 - x_2 \leq 3$

welches analog zur ersten Teilaufgabe gelöst werden kann. Die Eckpunkte sind:

- $x = (1, 0)$,
- $x = (3, 0)$,
- $x = (5, 2)$.

Der Simplex-Algorithmus beginnend bei $(1, 0)$ findet das Ergebnis direkt: $(1, 0) \to (5, 2)$. Das Ergebnis des originalen LPs ist somit $x = (5, 2, 1)$.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2019 Prof. R. Wattenhofer

Technische Informatik II

Klausur

Freitag, 23. August 2019, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. **Begründen Sie alle Ihre Antworten** sofern nichts anderes steht. Beschriften Sie Skizzen und Zeichnungen verständlich. **Was wir nicht lesen können gibt keine Punkte!**

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie jedes Zusatzblatt ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Familienname	**Vorname**	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - SQL		26
Summe		**26**

1 SQL (26 Punkte)

Gegeben ist eine SQL Datenbank mit folgenden Tabellen:

students(id, name, firstname, active)
registered(student_id, course_id, semester, year)
courses(id, title, credits, lecturer_id)
lecturers(id, name, firstname, academic_title, active)
prerequisites(course_id, prerequisite_id)

Die Primary Keys (oder Key-Paare) sind jeweils unterstrichen. Die Datenbank enthält Daten über alle Studierenden, Dozierenden und Vorlesungen an der ETH seit dem Jahr 1990. Die Tabelle *prerequisites* gibt vorausgesetzte Vorlesungen an. Zum Beispiel, bevor man TI2 belegen kann sollte man TI1 besucht haben.

Einträge in dieser Datenbank sehen zum Beispiel so aus:

students('16-909-384', 'Muster', 'Maxine', TRUE)
registered('16-909-384', '227-0014-00L', 'FS', '2019')
courses('227-0014-00L', 'TI2', 4, '19-691-117')
lecturers('19-691-117', 'Wattenhofer', 'Roger', 'Prof. Dr.', TRUE)
prerequisites('227-0014-00L', '227-0013-00L')

Schreiben Sie SQL Queries um die folgenden Fragen zu beantworten:

a) `[6 Punkte]` Geben Sie Name und Vorname aller Dozierenden aus, welche mindestens einen Doktortitel ('Dr.') haben.

b) `[6 Punkte]` Geben Sie für jede Vorlesung den Vorlesungstitel und die entsprechende Anzahl an vorausgesetzten Vorlesungen aus.

c) `[8 Punkte]` Finden Sie die Namen aller immatrikulierten (aktiven) Studenten, die Vorlesungen bei Dozierenden gehört haben, die bereits emeritiert wurden (nicht aktiv).

d) `[6 Punkte]` Im Folgenden ist eine SQL Query gegeben.

```
SELECT s.id, s.name, COUNT(r.course_id) AS cnt
FROM students s
LEFT OUTER JOIN registered r ON r.student_id = s.id
   AND r.year >= 2012 AND r.year <= 2019
GROUP BY s.id, s.name
ORDER BY cnt DESC
```

Füllen Sie in die unten vorgegebene leere Tabelle realistische Werte für die erste, zweite und letzte Zeile des Queryresultats ein. Begründen Sie ihre Wahl für den Wert von `cnt` in der ersten Zeile des Resultats.

⋮	⋮	⋮	⋮

Lösungen

a)
```
SELECT name, firstname
FROM lecturers
WHERE academic_title LIKE '%Dr.%'
```

b) Brauchen LEFT OUTER JOIN damit wir auch Vorlesungen ohne prerequisites ausgeben. Brauchen GROUP BY da nach dem LEFT OUTER JOIN für jede Vorlesung eine Zeile pro Voraussetzung da ist. Nach dem GROUP BY kommt jede Vorlesung noch einmal vor, und mit COUNT können wir dann für jede Vorlesung einfach die Anzahl Prerequisites zählen.

```
SELECT c.title, COUNT(p.prerequisite_id)
FROM courses AS c
LEFT OUTER JOIN prerequisites AS p ON c.id = p.course_id
GROUP BY c.title
```

c) Brauchen GROUP BY damit der gleiche Student nicht mehrmals ausgegeben wird. Müssen mit s.id gruppieren damit wir Studenten mit gleichen Vor- und Nachnamen separat auflisten.

```
SELECT s.firstname, s.name
FROM students AS s
JOIN registered AS r ON r.student_id = s.id
JOIN courses AS c ON c.id = r.course_id
JOIN lecturers AS l ON l.id = c.lecturer_id
WHERE s.active = TRUE AND l.active = FALSE
GROUP BY s.id
```

d) Die Query gibt alle IDs und Nachnamen der Studierenden aus, sowie die Anzahl der Kurse für die sie sich in den Jahren 2012 bis 2019 (jeweils inklusive) eingeschrieben haben. Studierende werden dabei höchstens einfach gelistet (wegen `GROUP BY`), Studierende werden aber auch mindestens einmal gelistet (wegen `LEFT OUTER JOIN`). Das Ergebnis ist nach der Anzahl der belegten Kurse in absteigender Reihenfolge geordnet. In der Query sind nur 3 Spalten gefragt, deshalb muss in der Tabelle auch eine Spalte frei bleiben. Es ist auch angegeben, dass die erste, zweite und letzte Zeile des Resultats angegeben werden soll, deswegen müssen alle drei Zeilen ausgefüllt sein. Die erste Spalte muss Legi-Nummern im korrekten Format enthalten, und die zweite Spalte muss Nachnamen enthalten.

Begründung für Wert von cnt in der ersten Zeile: Dieser Student hat zwischen 2012 und 2019 die meisten Vorlesungen besucht. Nehmen wir an, dass er/sie in 8 Jahren Bachelor und Master macht, dann sind 50 Vorlesungen sehr viel, aber noch realistisch.

13-567-874	Schmid	50	
12-875-698	Ruefer	49	
⋮	⋮	⋮	
08-453-789	Gerber	0	

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2019 Prof. R. Wattenhofer

Technische Informatik II / Betriebssysteme & Netzwerke

Klausur

Mittwoch, 22. Januar 2020, 09:00 - 10:30 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!
Lesen Sie die folgenden Anweisungen!

Die Prüfung dauert 90 Minuten und es gibt insgesamt 90 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. **Begründen Sie alle Ihre Antworten** sofern nichts anderes steht. Beschriften Sie Skizzen und Zeichnungen verständlich. **Was wir nicht lesen können gibt keine Punkte!**

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie jedes Zusatzblatt ebenfalls mit Ihrem Namen und Ihrer Legi-Nummer.

Familienname	**Vorname**	**Legi-Nr.**

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1 - Multiple Choice		3
2 - Databases and SQL		24
Summe		**27**

Name	**Legi-Nr.**

1 Multiple Choice (3 Punkte)

Geben Sie bei jeder Aussage an, ob sie wahr oder falsch ist. Jede korrekte Antwort gibt 1 Punkt. Jede fehlerhafte Antwort und unbeantwortete Aussage gibt 0 Punkte. **In dieser Aufgabe können Sie Ihre Antworten nicht begründen oder erklären.**

a) `[3 Punkte]` Gegeben ist folgende Menge von Schlüsseln:

$$K = \{11, 14, 16, 22, 27, 29, 33, 40, 42, 44\}.$$

Markieren Sie diejenigen Hashfunktionen mit „wahr“ (sonst: „falsch“), welche zu den wenigsten Kollisionen führen (bei Gleichheit bitte mehrere mit „wahr“ markieren):

	wahr	**falsch**
$h(k) = k \ mod \ 11$	☐	☐
$h(k) = k \ mod \ 12$	☐	☐
$h(k) = k \ mod \ 13$	☐	☐

Lösungen

1 Multiple Choice (3 Punkte)

a) [3 Punkte] Gegeben ist folgende Menge von Schlüsseln:

$$K = \{11, 14, 16, 22, 27, 29, 33, 40, 42, 44\}.$$

Markieren Sie diejenigen Hashfunktionen mit „wahr“ (sonst: „falsch“), welche zu den wenigsten Kollisionen führen (bei Gleichheit bitte mehrere mit „wahr“ markieren):

	wahr	**falsch**
$h(k) = k \bmod 11$ *Begründung:* $h(K) = \{0, 3, 5, 0, 5, 7, 0, 7, 9, 0\}$*, die Anzahl der Kollisionen ist also:* $\underbrace{\binom{4}{2}}_{Zelle\ 0} + \underbrace{\binom{2}{2}}_{Zelle\ 5} + \underbrace{\binom{2}{2}}_{Zelle\ 7} = 8.$		✓
$h(k) = k \bmod 12$ *Begründung:* $h(K) = \{11, 2, 4, 10, 3, 5, 9, 4, 6, 8\}$*, die Anzahl der Kollisionen ist also:* $\underbrace{\binom{2}{2}}_{Zelle\ 4} = 1.$	✓	
$h(k) = k \bmod 13$ *Begründung:* $h(K) = \{11, 1, 3, 9, 1, 3, 7, 1, 3, 5\}$*, die Anzahl der Kollisionen ist also:* $\underbrace{\binom{3}{2}}_{Zelle\ 1} + \underbrace{\binom{3}{2}}_{Zelle\ 3} = 6.$		✓

2 Databases and SQL (24 Punkte)

Gegeben ist eine SQL Datenbank mit folgenden Tabellen:

students(id, name, firstname, birthdate)
registered(student_id, course_id)
courses(id, title, examdate, credits, weekday, slot)

Die Tabellen enthalten die Daten der ETH für ein Semester. Unterstrichene Attribute markieren Primary Keys. Vorlesungen starten immer zur geraden Doppelstunde. Der Slot einer Vorlesung gibt die Uhrzeit einer Vorlesung an, zum Beispiel sind Vorlesungen im Slot 0 von 8:00 bis 10:00 Uhr.

a) `[4 Punkte]` Geben Sie in natürlicher Sprache an, was die folgende Anfrage abfragt:

```
SELECT firstname, name
FROM students
ORDER BY birthdate ASC
LIMIT 1
```

Schreiben Sie SQL Anfragen für folgende Fragestellungen

b) `[6 Punkte]` Bestimmen Sie die Vornamen und Namen der Studierenden, die mindestens eine Prüfung an Ihrem Geburtstag schreiben.

c) `[6 Punkte]` Geben Sie die eindeutigen Paare von Vorlesungs-IDs zurück von den Vorlesungen, die gleichzeitig stattfinden.

Name	Legi-Nr.

Zum Abschluss vergleichen wir verschiedene Methoden mehrere Tabellen zu kombinieren: `INNER JOIN`, `FULL OUTER JOIN` und `LEFT OUTER JOIN`.

d) `[2 Punkte]` Ordnen Sie die drei vorgestellten JOIN Techniken nach der Grösse ihrer Ergebnisse.

e) `[6 Punkte]` Gegeben sei das Schema mit den zwei Tabellen `T1(A, X)` und `T2(X, B)`. Wir wollen wiederum `INNER JOIN`, `FULL OUTER JOIN` und `LEFT OUTER JOIN` ausführen.

Befüllen Sie die beiden Tabellen mit so **wenig** Zeilen wie möglich, sodass alle Ergebnisse unterschiedlich viele Zeilen haben. Geben Sie ausserdem die Anfrageergebnisse an.

Die allgemeine Anfrage ist wie folgt (⋈ ist der Platzhalter für jeweils `INNER JOIN`, `FULL OUTER JOIN` oder `LEFT OUTER JOIN`):

```
SELECT T1.A, T2.B
FROM T1 ⋈ T2 ON T1.X=T2.X
```

T1.A	T1.X

T2.X	T2.B

INNER JOIN :

FULL OUTER JOIN :

LEFT OUTER JOIN :

Lösungen

2 Databases and SQL (24 Punkte)

a) Die Anfrage ermittelt den Namen des **ältesten** Studenten. (Keine Lösung: Die Anfrage sortiert die Studenten nach dem Alter und gibt davon ein Tupel zurück)

b)
```
SELECT DISTINCT students.name, students.firstname
FROM students
INNER JOIN registered ON students.id=registered.student_id
INNER JOIN courses ON registered.course_id=courses.id
WHERE students.birthdate = courses.examdate
```

c) Wir machen einen Selbstjoin auf die Tabelle `courses`. Dabei müssen wir verhindern, eine Vorlesung mit sich selbst und Paare von Vorlesungen doppelt zu vergleichen.

```
SELECT c1.id, c2.id
FROM courses c1
INNER JOIN courses c2
ON c1.id < c2.id
AND c1.weekday = c2.weekday
AND c1.slot = c2.slot
```

d) Die JOINS werden wie folgt in der Kardinalität grösser: `INNER JOIN` < `LEFT OUTER JOIN` < `FULL OUTER JOIN`.

e)

T1.A	T1.X
1	1

T2.X	T2.B
2	2

Diese erzeugen die folgenden Joinergebnisse:

INNER JOIN <leeres Ergebnis >

FULL OUTER JOIN (1,NULL),(NULL,2)

LEFT OUTER JOIN (1,NULL)