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 - Multiple Choice | | 7 |
| 2 - Projektplanung | | 25 |
| 3 - Concurrency | | 20 |
| 4 - Disks und Dateisysteme | | 18 |
| 5 - Security | | 20 |
| **Summe** | | **90** |

# 1 Multiple Choice (7 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 oder erklären.**

| **Aussage** | **wahr** | **falsch** |
|---|---|---|
| Die time-to-live (TTL) ermöglicht es, dringende Anfragen mit Priorität zu bearbeiten. | ☐ | ☐ |
| Nur Router benutzen eine Routingtabelle. | ☐ | ☐ |
| Routingtabellen beinhalten nur Einträge über die Nachbarn eines Knotens. | ☐ | ☐ |
| Wenn beim Aufrufen einer Webseite ein 5xx Error auftritt, heisst dies, der Client konnte sich nicht mit dem Netzwerk verbinden. | ☐ | ☐ |
| Ein DNS lookup kann sowohl einen Domainnamen in eine IP-Adresse übersetzen als auch eine IP-Adresse in einen Domainnamen. | ☐ | ☐ |
| HTML ist ein einfaches Protokoll zur Mail Übermittlung. | ☐ | ☐ |
| Das IMAP Protokoll erlaubt das Abrufen von E-Mails auf mehreren Geräten. | ☐ | ☐ |

**Lösungen**

| **Aussage** | **wahr** | **falsch** |
|---|---|---|
| Die time-to-live (TTL) ermöglicht es, dringende Anfragen mit Priorität zu bearbeiten.<br>*Begründung: Sie verhindert, dass Pakete endlos im Internet herumreisen.* | | ✓ |
| Nur Router benutzen eine Routingtabelle.<br>*Begründung: Jedes Gerät mit Netzwerkanschluss (eingeschlossen PCs) hat eine Routingtabelle, die einen Eintrag für localhost und für jedes (physische und logische) Netzwerkinterface enthält.* | | ✓ |
| Routingtabellen beinhalten nur Einträge über die Nachbarn eines Knotens.<br>*Begründung: Routingtabellen enthalten auch Einträge über Zielbereiche, mit denen dder Knoten nicht direkt verbunden ist, um Pakete passend weiterleiten zu können. Adressen werden nach Präfix groupiert um die Routingtabelle effizient verwalten zu können.* | | ✓ |
| Wenn beim Aufrufen einer Webseite ein 5xx Error auftritt, heisst dies, der Client konnte sich nicht mit dem Netzwerk verbinden.<br>*Begründung: Falsch, 5xx Errors sind Fehler auf Serverseite.* | | ✓ |
| Ein DNS lookup kann sowohl einen Domainnamen in eine IP-Adresse übersetzen als auch eine IP-Adresse in einen Domainnamen.<br>*Begründung: Ja, DNS funktioniert in beide Richtungen.* | ✓ | |
| HTML ist ein einfaches Protokoll zur Mail Übermittlung.<br>*Begründung: Nein, HTML ist eine Markup-Language, kein Protokoll.* | | ✓ |
| Das IMAP Protokoll erlaubt das Abrufen von E-Mails auf mehreren Geräten.<br>*Begründung: Ja, beim IMAP Protokoll werden die Nachrichten auf einem Server gespeichert, von wo aus sie von mehreren Clienten abgefragt werden können.* | ✓ | |

## 2 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.

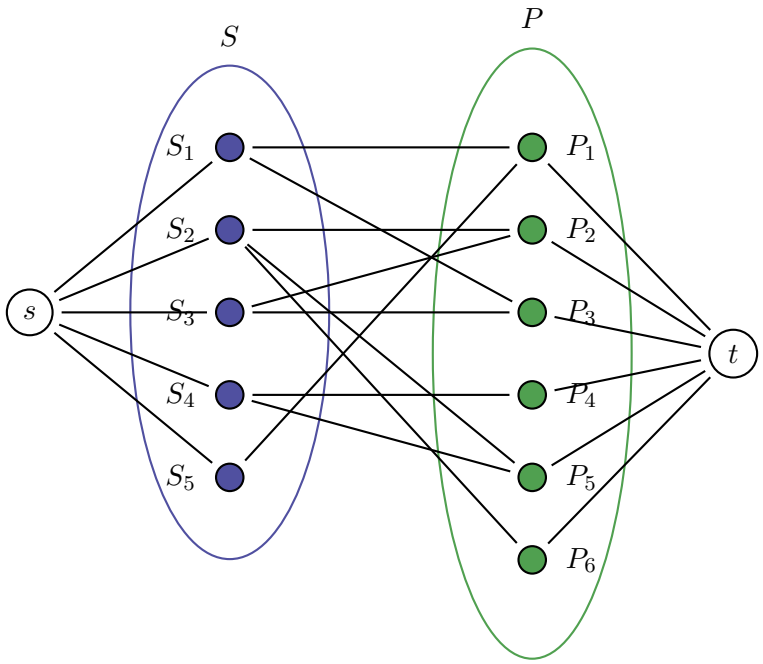


Abbildung 1: Netzwerk zu Aufgabe a)

**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.

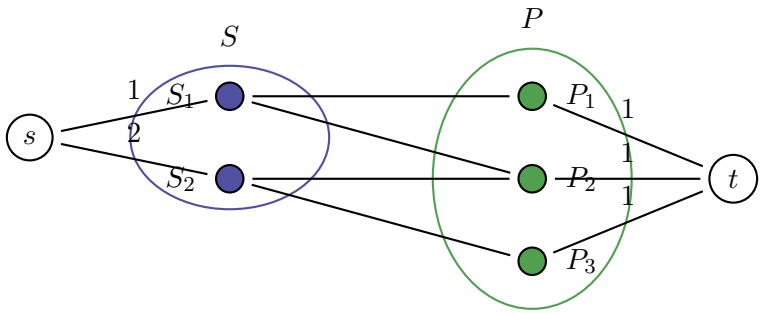


Abbildung 2: Beispiellösung d)

# 3 Concurrency (20 Punkte)

In einem High-Frequency-Trading-System muss eine grosse Zahl von Nachrichten **mit möglichst kleiner Verzögerung** verarbeitet werden. Um die Daten einzulesen soll eine Shared-Memory Implementierung mit einem Ringpuffer zur Anwedung kommen. Ein Writer-Prozess schreibt die Nachrichten in den Puffer und ein Reader-Prozess verarbeitet eine nach der anderen.

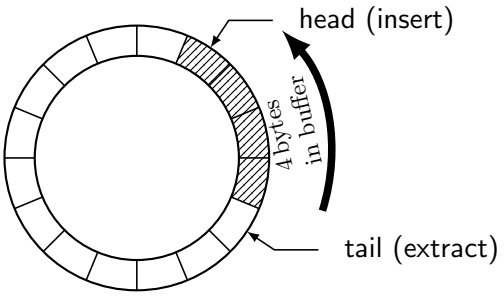


Abbildung 3: Ringpuffer

```
1:  procedure READER
2:      Message m
3:
4:      loop
5:
6:          while head = tail do
7:
8:
9:          end while
10:
11:         tail ← next(tail)
12:
13:         m ← buffer[tail]
14:
15:         process(m)
16:
17:     end loop
18:
19: end procedure
```

```
1:  procedure WRITER
2:      Message m
3:
4:      loop
5:
6:          m ← receive()
7:
8:          while next(head) = tail do
9:
10:
11:         end while
12:
13:         buffer[next(head)] ← m
14:
15:         head ← next(head)
16:
17:     end loop
18:
19: end procedure
```

**a)** [12] Der Pseudocode implementiert einen Ringpuffer. Jedoch ist die Implemetierung nicht thread-safe. Das heisst, die Korrektheit des Algorithmus ist nicht garantiert wenn der Code in mehreren Threads ausgeführt wird. Lösen Sie dieses Problem, indem Sie an geeigneten Stellen im Code das Lock L verwenden. Gehen Sie dabei davon aus, dass das Lesen oder Schreiben einer Variablen möglicherweise *nicht* atomar erfolgt. Sie können lock(L) und unlock(L) verwenden. Achten Sie darauf, dass das Lock jeweils nur so lange wie wirklich nötig gehalten wird.
Anmerkung: buffer, head, tail und L sind globale Objekte, werden also von beiden Threads geteilt.

**b)** [2] Bei einem Test stellt sich heraus, dass der Puffer regelmässig komplett voll ist, weil der Writer oft schneller Daten produziert als der Reader konsumiert. Sie kriegen den Rat, den Puffer zu vergrössern. Wieso löst dies das Problem generell nicht?

**c)** [2] Was ist ein weiteres Problem von grossen Puffern für den vorliegenden Anwendungsfall?

**d)** [4] Die Implementierung oben verwendet Spinning. Eine Alternative ist, dass der Writer-Thread dem Reader die Nachrichten mit Message Passing übermittelt. Welche Variante ist im vorliegenden Fall zu bevorzugen? Nehmen Sie an, dass das oben beschriebene System als einzige Anwendung auf einer Mehrkernmaschine läuft. Zur Erinnerung: Der Betriebssystemkern läuft in einem oder mehreren weiteren Prozess(en).

## Lösungen

**a)**

```
1:  procedure READER
2:      Message m
3:
4:      loop
5:          lock(L)
6:          while head = tail do
7:              unlock(L)
8:              lock(L)
9:          end while
10:
11:         tail ← next(tail)
12:         unlock(L)
13:         m ← buffer[tail]
14:
15:         process(m)
16:
17:     end loop
18:
19: end procedure
```

```
1:  procedure WRITER
2:      Message m
3:
4:      loop
5:
6:          m ← receive()
7:          lock(L)
8:          while next(head) = tail do
9:              unlock(L)
10:             lock(L)
11:         end while
12:         unlock(L)
13:         buffer[next(head)] ← m
14:         lock(L)
15:         head ← next(head)
16:         unlock(L)
17:     end loop
18:
19: end procedure
```

Erklärung: Jede Schreiboperation von *head* und *tail* muss gelockt werden. *tail* wird vom Reader geschrieben, und muss deshalb beim Writer für jede Operation (auch lesen) gelockt werden. Umgekehrt gilt dasselbe für *head*. Die beiden unlock-lock Operationen in der while-Schleife sind nötig, um einen Deadlock zu verhindern.

**b)** Das Problem kann so nicht gelöst werden. Da der Writer schneller Daten produziert als konsumiert werden, wird der Puffer trotzdem noch überlaufen, einfach später.

**c)** Die Latenzzeit wird erhöht, wenn der Puffer mehr gefüllt ist.

**d)** Spinning (in den beiden **while**-Schleifen) kann zu hohem CPU load führen, ohne dass tatsächlich Arbeit verrichtet wird. Mit nur zwei Threads wird das aber kein Problem sein. Message Passing geht über den Kernel und braucht deshalb zwei Kontextswitches, um eine Nachricht zu übermitteln. (OSYS, Section 3.4: “Shared memory is faster than message passing, as message passing systems are typically implemented using system calls and thus require the more time-consuming task of kernel intervention.”) Deshalb bleibt weniger Zeit, um den Algorithmus der Applikation auszuführen. Die Performance wird also schlechter sein. Grundsätzlich ist die Performance (insbesondere die Latenz) von Spinning optimal, so lange weniger Threads als Kerne vorhanden sind.

# 4 Disks und Dateisysteme (18 Punkte)

Angenommen wir haben eine HDD, auf der ein inode-basiertes Dateisystem gespeichert ist. Das Dateisystem lässt ein Maximum von 10 Dateien zu, hat 15 Datenblöcke mit einer Datenblockgrösse von 1MB, und enthält folgende Dateien:

- `/mov/EOT` 3 MB gross
- `/mov/GHD` 2.5 MB gross
- `/mov/home/movie` 0.5 MB gross
- `/tools/vlc` 0.1 MB gross

**a)** `[14]` Vervollständigen Sie in Tabelle 4 die Belegung der Blöcke unserer HDD. Tabelle 3 gibt vor, wie die Bitmaps, inode-Blöcke und Datenblöcke nummeriert sind. In einen Datenblock, der den Inhalt der Datei X enthält, schreiben Sie bitte X. Wenn eine Datei X auf mehrere Datenblöcke aufgeteilt wird, benennen Sie die einzelnen Teile bitte $X_1$, $X_2$ usw.

Verwenden Sie Tabelle 5 als Ersatz, falls Sie Fehler beim Ausfüllen der Lösung in Tabelle 4 machen. Markieren Sie klar, welche Lösung gültig ist.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Super | 0 | 1 | 0 | 1 | 2 | 0 | 1 | 0 | 1 | 2 | 3 | 4 |
| | 2 | 3 | 3 | 4 | 5 | 2 | 3 | | | | | |
| | 4 | 5 | 6 | 7 | 8 | 4 | 5 | | | | | |
| | 6 | 7 | 9 | 10 | 11 | 6 | 7 | | | | | |
| | 8 | 9 | 12 | 13 | 14 | 8 | 9 | | | | | |
| | i-bmap | | d-bmap | | | inodes | | data | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| data | | | | | | | | | |

Tabelle 3: Beispiellayout mit nummerierten Zellen.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Super | 1 | | 1 | 1 | | 0 | | mov:4<br>tools:8 | vlc:3<br>zip:1 | | vlc | zip |
| | | | | | | | | | | | | |
| | | | | | | 2 | 9,7,5 | | | | | |
| | 0 | 1 | | | | | 11,8,6 | | | | | |
| | | | | 0 | 0 | 1 | 12 | | | | | |
| | i-bmap | | d-bmap | | | inodes | | data | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $EOT_1$ | | | movie:2<br>zip:1 | | |
| data | | | | | | | | | |

Tabelle 4: Layout zum Eintragen der Lösung.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Super | 1 | | 1 | 1 | | 0 | | mov:4<br>tools:8 | vlc:3<br>zip:1 | | vlc | zip |
| | | | | | | | | | | | | |
| | | | | | | 2 | 9,7,5 | | | | | |
| | 0 | 1 | | | | | 11,8,6 | | | | | |
| | | | | 0 | 0 | 1 | 12 | | | | | |
| | i-bmap | | d-bmap | | | inodes | | | | data | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $EOT_1$ | | | movie:2<br>zip:1 | | |
| | | | | data | | | | | |

Tabelle 5: Layout zum Eintragen der Lösung.

**b)** [4] Welche weiteren Dateien sind zusätzlich zu den oben aufgelisteten auch auf der HDD gespeichert? Geben Sie den vollständigen Pfad an.

**Lösungen**

**a)** Lösung in Tabelle 6

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Super | 1 | 1 | 1 | 1 | 1 | 0 | 4 | mov:4<br>tools:8 | vlc:3<br>zip:1 | EOT:5<br>GHD:7<br>home:9 | vlc | zip |
| | 1 | 1 | 1 | 1 | 1 | 10 | 3 | | | | | |
| | 1 | 1 | 1 | 1 | 1 | 2 | 9,7,5 | | | | | |
| | 0 | 1 | 1 | 1 | 1 | | 11,8,6 | | | | | |
| | 1 | 1 | 1 | 0 | 0 | 1 | 12 | | | | | |
| | i-bmap | | d-bmap | | | inodes | | | | data | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $EOT_3$ | $GHD_3$ | $EOT_2$ | $GHD_2$ | $EOT_1$ | movie | $GHD_1$ | movie:2<br>zip:1 | | |
| data | | | | | | | | | |

Tabelle 6: Lösungen.

**b)** Zusätzliche Dateien:

- `/mov/home/zip`
- `/tools/zip`

## 5 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.