PRAKTISCHE INFORMATIK-KOMPETENZEN INTERAKTIV AN LAPTOPS PRÜFEN

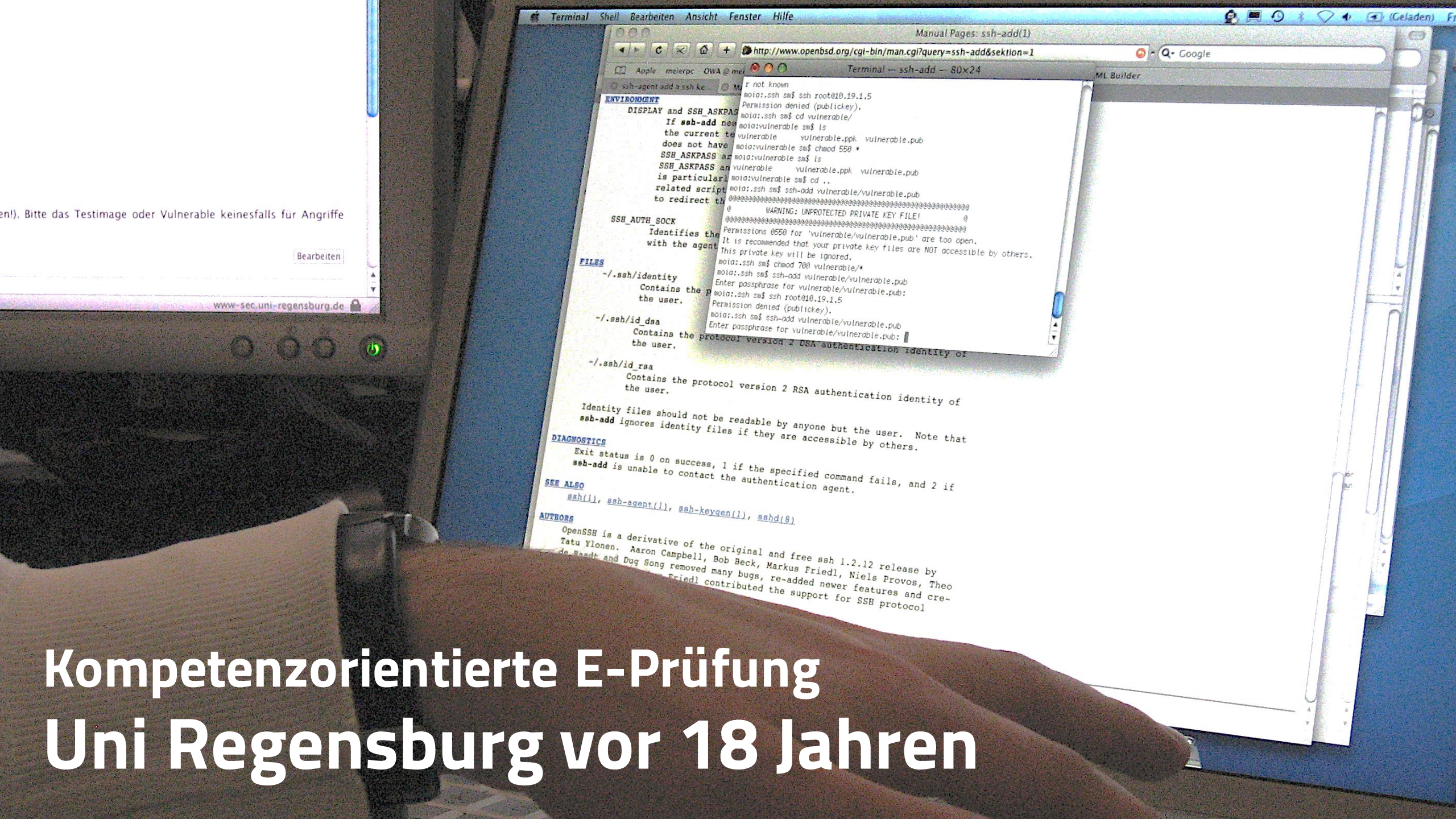
Ein resilientes System für große E-Prüfungen bei kleinen Ressourcen

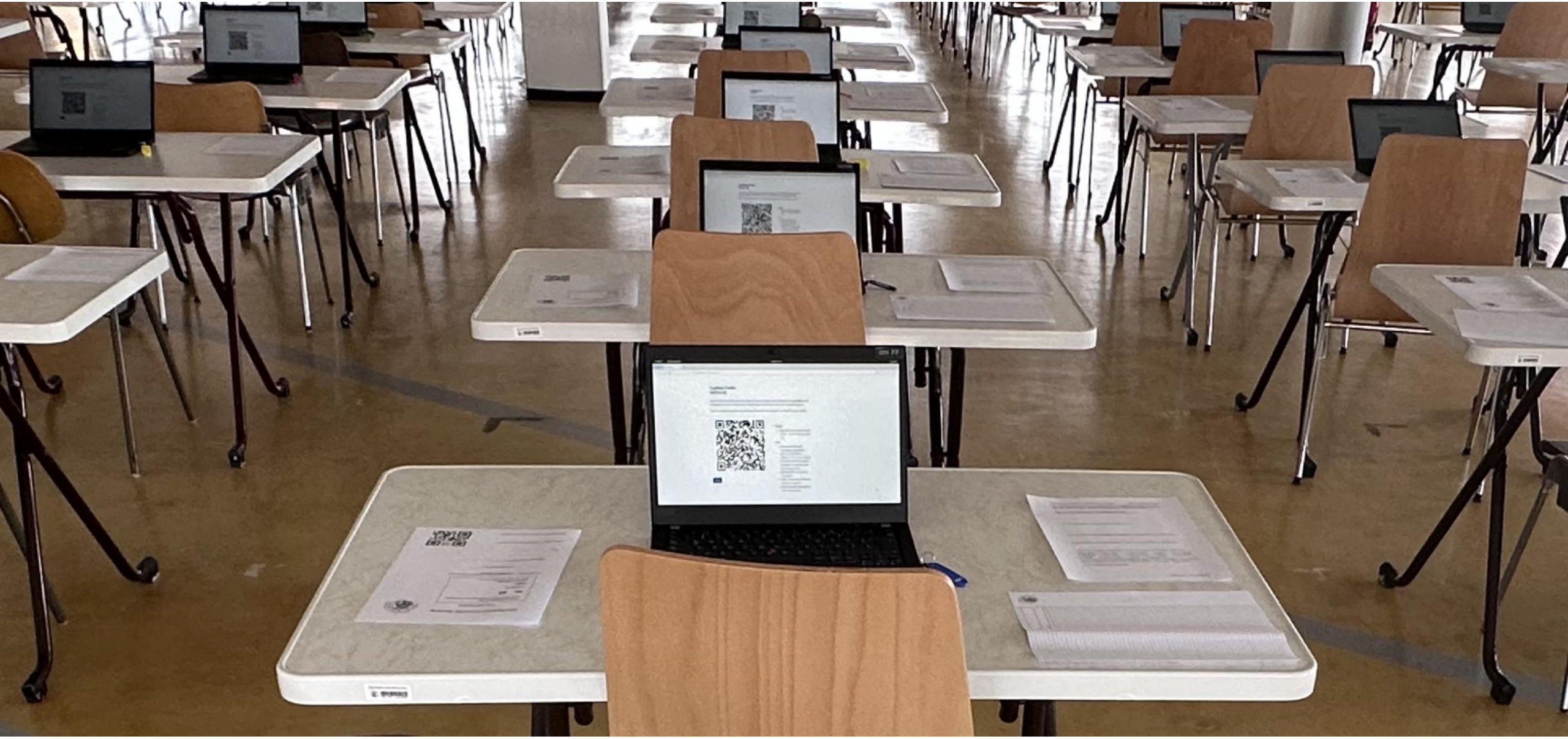
Prof. Dr. **Dominik Herrmann**

@herdom · herdom.net

LSt Privatsphäre und Sicherheit in Informationssystemen (PSI) an der Universität Bamberg

psi-exam


Kompetenzorientierte E-Prüfung
Uni Regensburg vor 18 Jahren



Vision: **einfacher e-prüfen in Präsenz (und zu Hause)**

AUSGANGSLAGE

Wunsch: Informatik-Prüfungen am Rechner (keine Handschrift)

Programmierfragen (Linux-Bash, Assembly, C, Python)
Reverse-Engineering von Programmen mit Ghidra
SQL-Injection-Angriffe auf Webanwendungen
Datenverkehrsanalyse (tcpdump/Wireshark)

Kompetenzorientiert prüfen mit beliebigen Fachanwendungen – **ohne Eingriff in private Rechner**

Unterstützung von Fernprüfungen gem. BayFEV

„**termingleiche Präsenzprüfung**“ … „unter strenger Beachtung der Grundsätze der **Chancengleichheit**“


Stream Chat
Send
Health
Stamina
250
/250
Teams
Leaderboard
Objectives
Primary weapon
YOUR COMMUNITY
VIP members
JackieTheMonster
Mikesparta
youcantseeme
alberon332
123XcreamX321
Dilimanko999
Members

AUSGANGSLAGE

Lösungen für E-Prüfungen (Auswahl)

PC-Pools: Dynexite (RWTH) · YAPS (TUHH), Examuntu (HAW HH) · Uni Regensburg (ILIAS)

BYOD: Safe Exam Environment (Klagenfurt) · PePP (BaWü)

TUMexam (handschriftl.)

TUxamine (TUB, Laptops, Moodle)

PC-Pools ungeeignet

zu geringe (und weiter sinkende) Kapazität,
Nutzungsrechte **schwer einschränkbar**,
hoher **Koordinationsaufwand**

Einschränkungen großer Prüfungsräume

kein zuverlässiges **Netz**, kein **Strom**
keine **dauerhafte Installation** möglich

Wunsch: flexible Klausuren mit neuen didaktischen Konzepten

EIGENSCHAFTEN VON PSI-EXAM

Große Prüfungen **ohne Netz/Strom,** ggf. videoüberwachte **Fernprüfung**

Prüfungen ausschließlich auf von uns ausgegebenen **Linux-Laptops**

Sicher, ausfallsicher **und** komfortabel

Technik **einfach** verständlich

Paperwork und Prozesse so wie bei Papierklausuren – dadurch **verwaltungskompatibel**

Minimale Abhängigkeiten zu Software-Bibliotheken/Plugins, dadurch langfristig mit wenig Aufwand betreibbar.

HARDWARE

Ergonomie-Anforderungen und
wartungsarme Bereitstellung

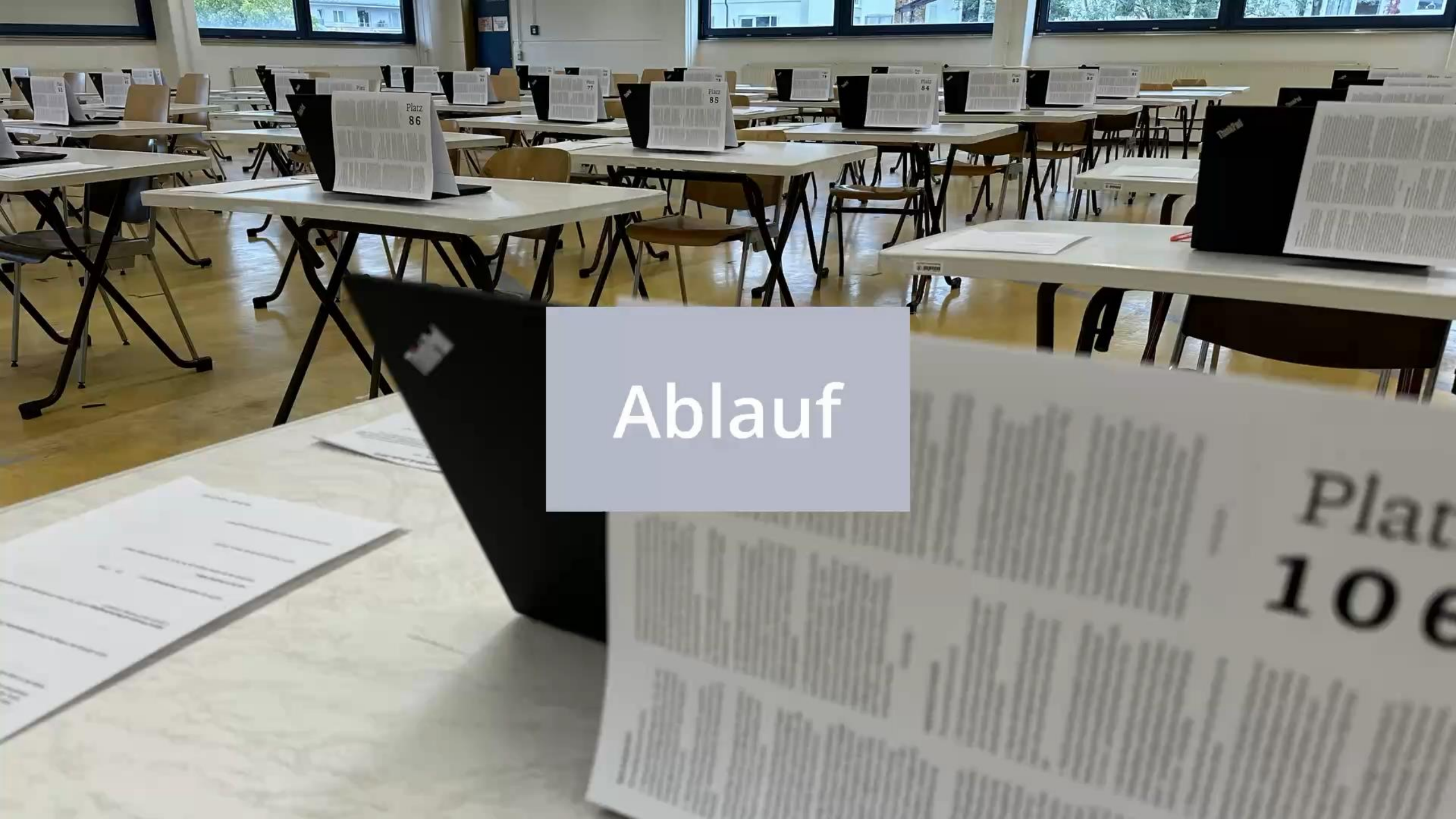
340 Laptops (Lenovo T14)
14“ mit **400 nits**, Core i5 2,4 GHz,
50 Wh Akku, Laden über USB-C

Privacy-Filter (upscreen), Euroboxen 40x30cm, Aruba AP-505 Access Points

Software: Debian, Gnome 42, Firefox

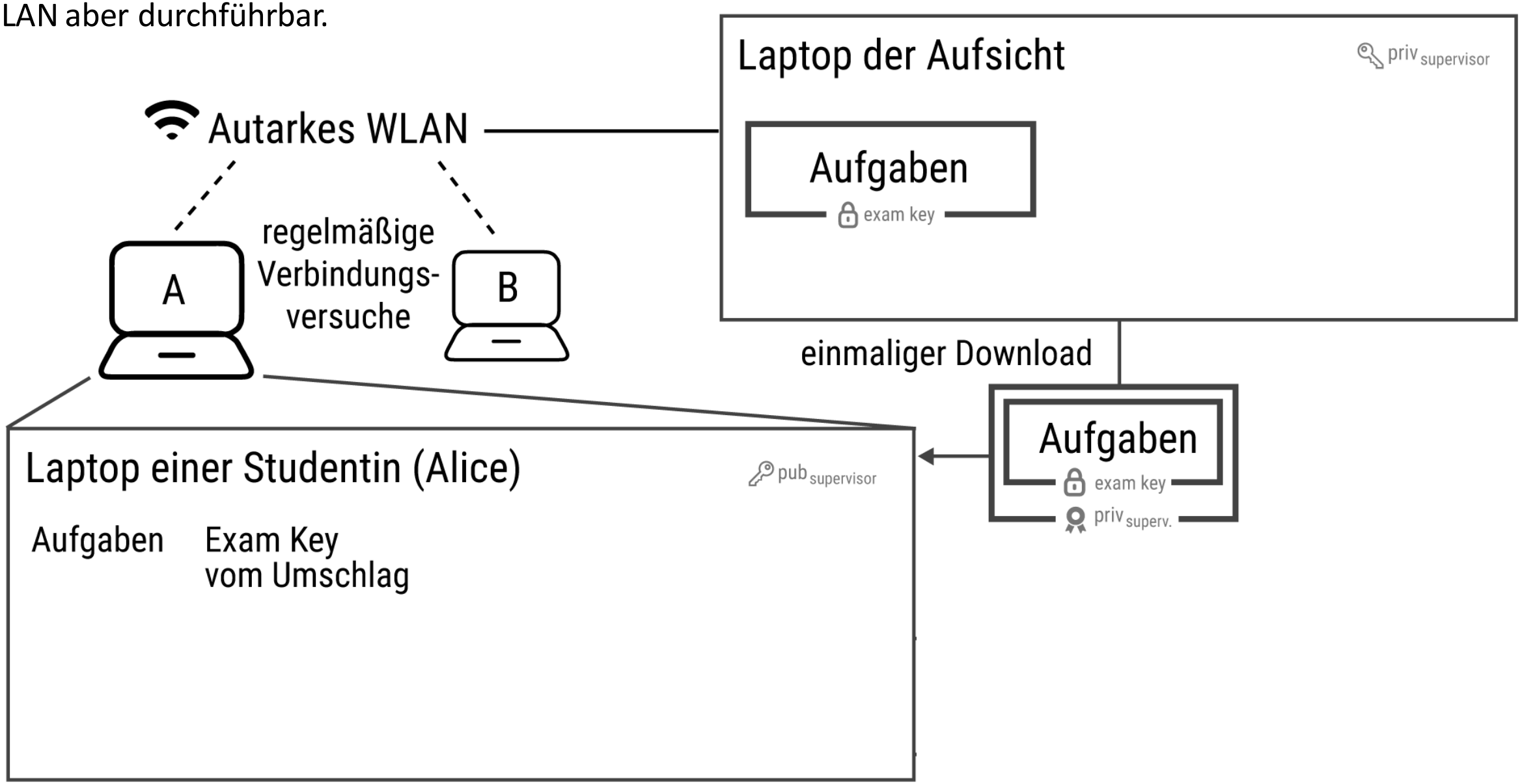
Softwareverteilung via Multicast mit Clonezilla (PXE-Boot) an 48-Port-Switch

Laden: an zwei IP-Steckdosen (GUDE 8041-1) mit Einschaltverzögerung (Last jew. 48 · 65 W = 3120 W bzw. 14 A).



Ablauf

Prüfungsdurchführung durch WLAN komfortabler;
bleibt bei gestörtem oder kompromittiertem
WLAN aber durchführbar.



LESSONS LEARNED NACH DREI PRÜFUNGEN

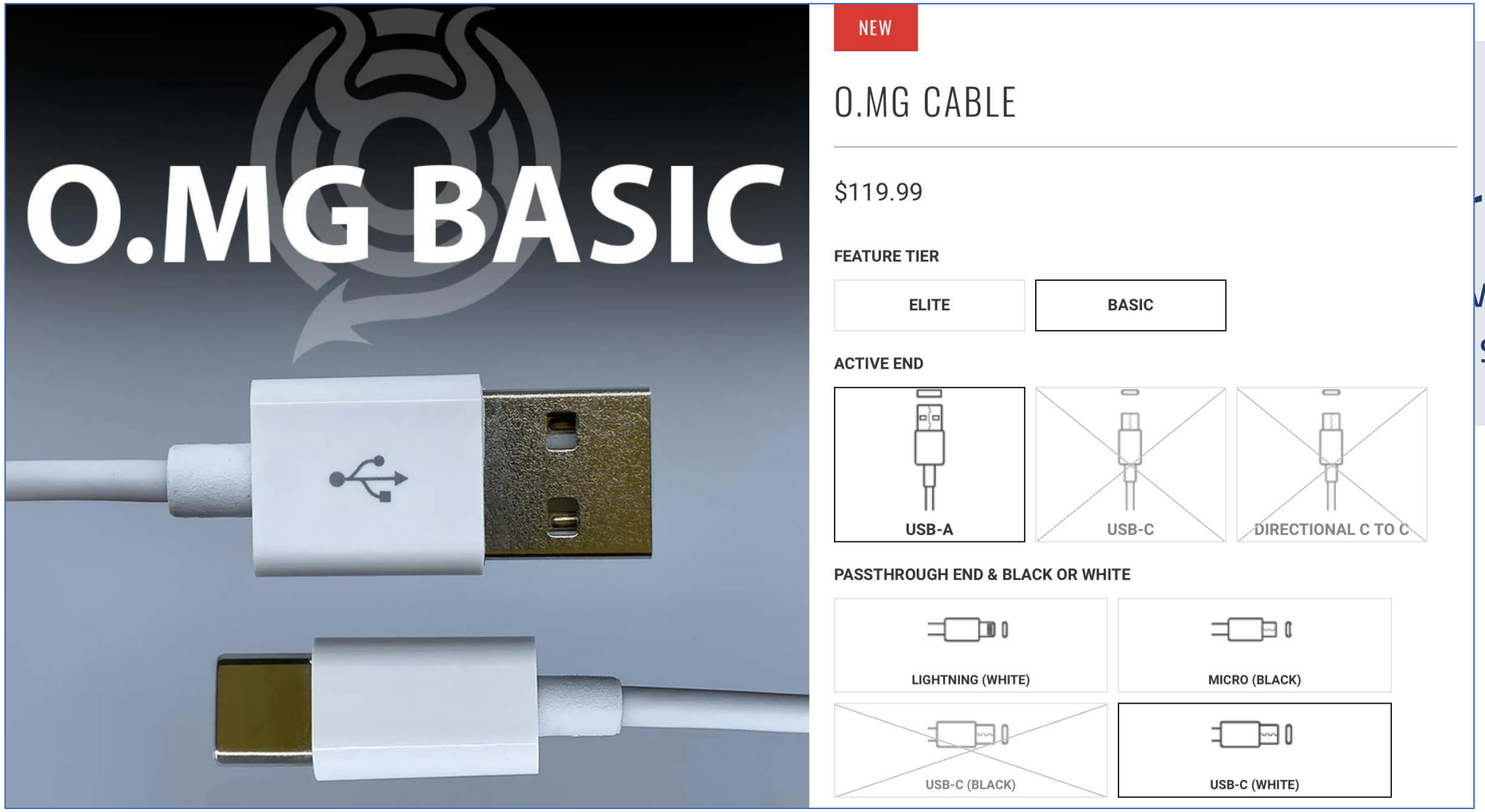
Probeklausur (n = 155)

1. Modulprüfung (n = 242)
2. Modulprüfung (n = 142)

Probleme mit **Linux**

Videoanleitung/Probeklausur hilfreich

Touchpad/Tastatur ungewohnt
Zukünftig Mäuse/Tastaturen stellen?



hilfreich

wohnt

stellen?

Verhält sich wie Tastatur, kann vorher aufgenommene Texte eingeben und Eingaben aufnehmen und via WiFi übertragen.

LESSONS LEARNED NACH DREI PRÜFUNGEN

Probeklausur (n = 155)

1. Modulprüfung (n = 242)
2. Modulprüfung (n = 142)

Probleme mit **Linux**

Videoanleitung/Probeklausur hilfreich

Touchpad/Tastatur ungewohnt
Zukünftig Mäuse/Tastaturen stellen?

Bearbeitung **langsamer** als auf Papier,
höheres Risiko sich zu verzetteln.

Im Vergleich zu gedruckten A3-Bögen
Aufgabenstellung **weniger übersichtlich**

Aufgabe 4: Assembly-Programmierung (21 Punkte)

Links ist das Assembly-Programm CIE angegeben, das auf dem Ihnen bekannten D-CORE (16-Bit Wortbreite, Big Endian) ausgeführt wird. Der rechts abgebildete Speicherinhalt zeigt die Situation vor der Ausführung des Programms. *Hinweis: 256_{10} entspricht 100_{16}.*

Am Ende dieser Aufgaben finden Sie die Befehle für den D-CORE-Prozessor.

```
 1 movi r0, 1
 2 lsli r0, 8
 3 movi r1, 0
 4 movi r2, 0
 5 ldw r3, r0

 6 loop:
 7 cmpeq r3, r1
 8 bt end

 9 addi r0, 2
10 ldw r4, r0
11 subi r3, 1
12 addu r2, r4
13 br loop

14 end:
15 mov r4, r2
16 bseti r2, 0
17 subu r2, r4
18 halt
```

```
OFFSET    SPEICHERINHALT
[...]
0x00f0:   00 00 00 00 00 00 00 00
0x00f8:   00 00 00 00 00 00 00 00
0x0100:   00 02 00 0e 00 1f 00 00
0x0108:   00 00 00 00 00 00 00 00
0x0110:   00 00 00 00 00 00 00 00
[...]
```

a) Ein anderes Programm enthält die Instruktion 0x3550. Verändern Sie diese Instruktion, sodass anstelle der ursprünglichen Operation – bei sonst gleichen Parametern – nun eine Subtraktion durchgeführt wird.

8 Punkte

☐ Weiß ich nicht.

Speichern

Geben Sie die Zeilennummern-Paare eines Use-After-Load-Hazards in CIE an.

Speichern

Zeigen Sie an einem Use-After-Load-Hazard, wie durch Umordnen der betroffenen Instruktionen der Hazard vermieden wird.

Speichern

Gibt es in CIE Control-Hazards? Geben Sie *nein* an, falls es *keine* Control-Hazards gibt. Andernfalls geben Sie *ein* Zeilennummer-Paar für einen Control-Hazard in CIE an.

Speichern

b) Welche Werte haben folgende Register, wenn CIE bis *einschließlich* Zeile 15 ausgeführt worden ist?

7 Punkte

☐ Weiß ich nicht.

Wert für Register **r0**:

Speichern

Wert für Register **r1**:

Speichern

Wert für Register **r2**:

Speichern

Wert für Register **r3**:

Speichern

c) In Zeile 18 enthält r2 entweder den Wert 0 oder 1. Welche Eigenschaft von r4 wird damit kodiert?

6 Punkte

☐ Weiß ich nicht.

Speichern

CIE soll nun auf einer Rechnerachitektur ausgeführt werden, in der es die Instruktion bseti nicht gibt. Daher muss die Funktionalität der Zeilen 15–17 ohne bseti realisiert werden. Vervollständigen Sie dazu das folgende Gerüst von CIE mit geeigneten Instruktionen und den dazugehörigen Parametern.

```
[...] ← r2 enthält das Zwischenergebnis
end:
```

Geben Sie hier Ihren Code ein ...

Speichern

```
halt
```

Assembly-Referenz: Befehle des D-CORE-Prozessors

Hier finden Sie alle Befehle des D-Core-Prozessors. `R[x]` steht für den Wert im Register x, `C` für den Wert des 1-Bit-Carry-Registers. `MEM[R[y]]` referenziert ein Wort an einer Speicheradresse, die in `R[y]` steht. `PC` ist der Program Counter.

Mnemonic	Kodierung	Hex	Bedeutung
			ALU-Operationen: ***Mnemonic x, y*** *Werte werden vorzeichenlos interpretiert.* *Nur addc ändert C: bei Überlauf C=1, sonst C=0.*
mov	0010 0000 <yyyy> <xxxx>	20yx	`R[x] = R[y]`
addu	0010 0001 <yyyy> <xxxx>	21yx	`R[x] = R[x] + R[y]`
addc	0010 0010 <yyyy> <xxxx>	22yx	`R[x] = R[x] + R[y] + C`
subu	0010 0011 <yyyy> <xxxx>	23yx	`R[x] = R[x] - R[y]`
and	0010 0100 <yyyy> <xxxx>	24yx	`R[x] = R[x] AND R[y]`
or	0010 0101 <yyyy> <xxxx>	25yx	`R[x] = R[x] OR R[y]`
xor	0010 0110 <yyyy> <xxxx>	26yx	`R[x] = R[x] XOR R[y]`
not	0010 0111 <****> <xxxx>	27*x	`R[x] = NOT R[x]`
			Vergleichs-Operationen: ***Mnemonic x, y***
cmpeq	0011 0000 <yyyy> <xxxx>	30yx	`C = (R[x] == R[Y])`
cmpne	0011 0001 <yyyy> <xxxx>	31yx	`C = (R[x] != R[Y])`
cmpgt	0011 0010 <yyyy> <xxxx>	32yx	`C = (R[x] > R[Y])`
cmplt	0011 0011 <yyyy> <xxxx>	33yx	`C = (R[x] < R[Y])`
			Immediate-Operationen: ***Mnemonic x, c*** *Diese Instruktionen verändern C nicht.*
movi	0011 0100 <cccc> <xxxx>	34cx	`R[X] = 0x000<c>`
addi	0011 0101 <cccc> <xxxx>	35cx	`R[X] = R[X] + 0x000<c>`
subi	0011 0110 <cccc> <xxxx>	36cx	`R[X] = R[X] - 0x000<c>`
andi	0011 0111 <cccc> <xxxx>	37cx	`R[X] = R[X] AND 0x000<c>`
lsli	0011 1000 <cccc> <xxxx>	38cx	`R[X] = R[X] << 0x000<c>`
lsri	0011 1001 <cccc> <xxxx>	39cx	`R[X] = R[X] >> 0x000<c>`
bseti	0011 1010 <cccc> <xxxx>	3Acx	`R[X] = R[X] OR (1 << 0x000<c>)`
bclri	0011 1010 <cccc> <xxxx>	3Bcx	`R[X] = R[X] AND NOT(1 << 0x000<c>)`
			Speicher-Operationen: ***Mnemonic x, y, c*** *Diese Instruktionen verändern C nicht.*
ldw	0100 <cccc> <yyyy> <xxxx>	4cyx	`R[X] = MEM[R[Y] + (0x000<c> << 1)]`
stw	0101 <cccc> <yyyy> <xxxx>	5cyx	`MEM[R[Y] + (0x000<c> << 1)] = R[X]`
			Kontrollfluss: ***Mnemonic label/imm12*** *PC bezieht sich auf PC vor dem Inkrementieren für den nächsten Befehl.* *Diese Instruktionen verändern C nicht.* *imm12 wird als vorzeichenbehafteter 12-Bit-Immediate-Wert im 2-Komplement interpretiert.*
br	1000 <iiii> <iiii> <iiii>	8iii	`PC = PC+2+<imm12>`
call	1001 <iiii> <iiii> <iiii>	9iii	`R[15] = PC+2; PC = PC+2+<imm12>`
bt	1010 <iiii> <iiii> <iiii>	Aiii	`(C == 1) ? PC = PC+2+<imm12> : PC = PC+2`
bf	1011 <iiii> <iiii> <iiii>	Biii	`(C == 0) ? PC = PC+2+<imm12> : PC = PC+2`
jmp	1100 <****> <****> <xxxx>	C**x	`PC = R[x]`

Ende von Aufgabe 4 / Ende der Klausur

Aufgabe 4 Assembly-Programmierung (21 P.)

Nebenstehend ist das Assembly-Programm CIE abgedruckt, das auf dem Ihnen bekannten D-CORE (16-Bit Wortbreite, Big Endian) ausgeführt wird. Der abgebildete Speicherinhalt zeigt die Situation *vor* der Ausführung des Programms. *Hinweis:* 256_{10} entspricht 100_{16}.

```
OFFSET    SPEICHERINHALT
[...]
0x00f0:   00 00 00 00 00 00 00 00
0x00f8:   00 00 00 00 00 00 00 00
0x0100:   00 02 00 0e 00 1f 00 00
0x0108:   00 00 00 00 00 00 00 00
0x0110:   00 00 00 00 00 00 00 00
[...]
```

```
1  movi r0, 1
2  lsli r0, 8
3  movi r1, 0
4  movi r2, 0
5  ldw r3, r0
6  loop:
7  cmpeq r3, r1
8  bt end
9  addi r0, 2
10 ldw r4, r0
11 subi r3, 1
12 addu r2, r4
13 br loop
14 end:
15 mov r4, r2
16 bseti r2, 0
17 subu r2, r4
18 halt
```

__ / 8 ☐ *Weiß ich nicht.*

a) Ein anderes Programm enthält die Instruktion `0x3550`. Verändern Sie diese Instruktion, sodass anstelle der ursprünglichen Operation – bei sonst gleichen Parametern – nun eine Subtraktion durchgeführt wird.

_____

Geben Sie die Zeilennummern-Paare *eines* Use-After-Load-Hazards in CIE an.

_____

Zeigen Sie an *einem* Use-After-Load-Hazard, wie durch Umordnen der betroffenen Instruktionen der Hazard vermieden wird.

_____

Gibt es in CIE Control-Hazards?

☐ nein

☐ ja, und zwar im Zeilennummer-Paar (*ein* Beispiel reicht): _____

__ / 7 ☐ *Weiß ich nicht.*

b) Welche Werte haben folgende Register, wenn CIE bis *einschließlich* Zeile 15 ausgeführt worden ist?

`r0`: _____ `r1`: _____ `r2`: _____ `r3`: _____

__ / 6 ☐ *Weiß ich nicht.*

c) In Zeile 18 enthält `r2` entweder den Wert 0 oder 1. Welche Eigenschaft von `r4` wird damit kodiert?

CIE soll nun auf einer Rechnerachitektur ausgeführt werden, in der es die Instruktion `bseti` nicht gibt. Daher muss die Funktionalität der Zeilen 15–17 *ohne bseti* realisiert werden. Vervollständigen Sie dazu das folgende Gerüst von CIE mit geeigneten Instruktionen und den dazugehörigen Parametern.

```
[...] ← r2 enthält das Zwischenergebnis
end:
```

Platz für Notizen (wird nicht bewertet):

Assembly-Referenz. `R[x]` steht für den Wert im Register `x`, `C` für den Wert des 1-Bit-Carry-Registers. `MEM[R[y]]` referenziert ein Wort an einer Speicheradresse, die in `R[y]` steht. `PC` ist der Program Counter.

ALU-Instruktionen

Werte werden vorzeichenlos interpretiert.
Nur `addc` ändert `C`: bei Überlauf `C=1`, sonst `C=0`.

Befehl/Kodierung	Syntax: *Befehl x, y*
mov 20yx	`R[x] = R[y]`
addu 21yx	`R[x] = R[x] + R[y]`
subu 23yx	`R[x] = R[x] - R[y]`
and 24yx	`R[x] = R[x] AND R[y]`
or 25yx	`R[x] = R[x] OR R[y]`
xor 26yx	`R[x] = R[x] XOR R[y]`
not 27*x	`R[x] = NOT R[x]`

Vergleichs-Instruktionen

Befehl/Kodierung	Syntax: *Befehl x, y*)
cmpeq 30yx	`C = (R[x] == R[Y])`
cmpne 31yx	`C = (R[x] != R[Y])`

Bei den Kontrollfluss-Instruktionen wird `imm12` als vorzeichenbehafteter 12-Bit-Immediate-Wert im 2-Komplement interpretiert.

Immediate- und Speicher-Instruktionen

Diese Instruktionen verändern `C` nicht.

Befehl/Kodierung	Syntax: *Befehl x, c* bzw. *Befehl x, y, c*
movi 34cx	`R[X] = 0x000<c>`
addi 35cx	`R[X] = R[X] + 0x000<c>`
subi 36cx	`R[X] = R[X] - 0x000<c>`
andi 37cx	`R[X] = R[X] AND 0x000<c>`
lsli 38cx	`R[X] = R[X] << 0x000<c>`
lsri 39cx	`R[X] = R[X] >> 0x000<c>`
bseti 3Acx	`R[X] = R[X] OR (1 << 0x000<c>)`
ldw 4cyx	`R[X] = MEM[R[Y] + (0x000<c> << 1)]`

Kontrollfluss-Instruktionen

PC bezieht sich auf PC *vor* dem Inkrementieren für den nächsten Befehl. Diese Instruktionen verändern C nicht.

Befehl/Kodierung	Syntax: *Befehl label/imm12*)
br 8iii	`PC = PC+2+<imm12>`
call 9iii	`R[15] = PC+2; PC = PC+2+<imm12>`
bt Aiii	`(C == 1) ? PC = PC+2+<imm12> : PC = PC+2`

Platz zur Bearbeitung beliebiger Aufgaben oder für Notizen; bitte bearbeitete Aufgabe angeben!

BEFRAGUNG

N = 127 Studierende

5% gaben an, dass sie ein **technisches** Problem hatten

4% gaben an, dass sie Probleme hatten, den **organisatorischen Ablauf** zu verstehen

Hilfsmittel

86% nutzten Taschenrechner
70% nutzten Linux-Konsole
27% nutzten Webseiten
22% nutzten Texteditor

81% „Live-Feedback während Bearbeitung war hilfreich“

```
<section>
  <p>a) Der fiktive Prozessor P22 arbeitet mit einer Taktrate von 2,2 MHz. Er verwendet keine Pipeline.
    Zur Abarbeitung von Instruktionen benötigt der P22 entweder einen, zwei, vier oder sechs Clock Cycles.
    Instruktionen der Gruppe IG-1 werden in <em>einem</em> Clock Cycle abgearbeitet.
    Instruktionen der Gruppe IG-2 benötigen hingegen <em>zwei</em> Clock Cycles, Instruktionen der
    Gruppe IG-4 <em>vier</em> Clock Cycles und Instruktionen der Gruppe IG-6 <em>sechs</em> Clock Cycles.
    Zu jedem Zeitpunkt wird höchstens eine Instruktion auf dem P22 ausgeführt.
  </p>
  <p>
    Ein Programm bestehe aus den folgenden Instruktionen, die alle abgearbeitet werden müssen:
    erstens 8,1 · 10<sup>7</sup> Instruktionen aus IG-2 und
    zweitens 3,7 · 10<sup>6</sup> Instruktionen aus Gruppe IG-4.
  </p>
  <p>
    Wie viele Minuten benötigt ein P22 im Idealfall für die Ausführung des Programms?
    <em>Rechenweg erforderlich. Runden Sie das Ergebnis auf zwei Nachkommastellen.</em>
  </p>
  <p>
    {{ task_input("2a_calculation", multiline=True, cols=50, rows=8) }}
  </p>
</section>

<aside>
  <strong>4 Punkte</strong>
  {{ i_dont_know_box("task2a") }}
</aside>
```

AUFGABENERSTELLUNG
VORERST MIT HTML/CSS/JS

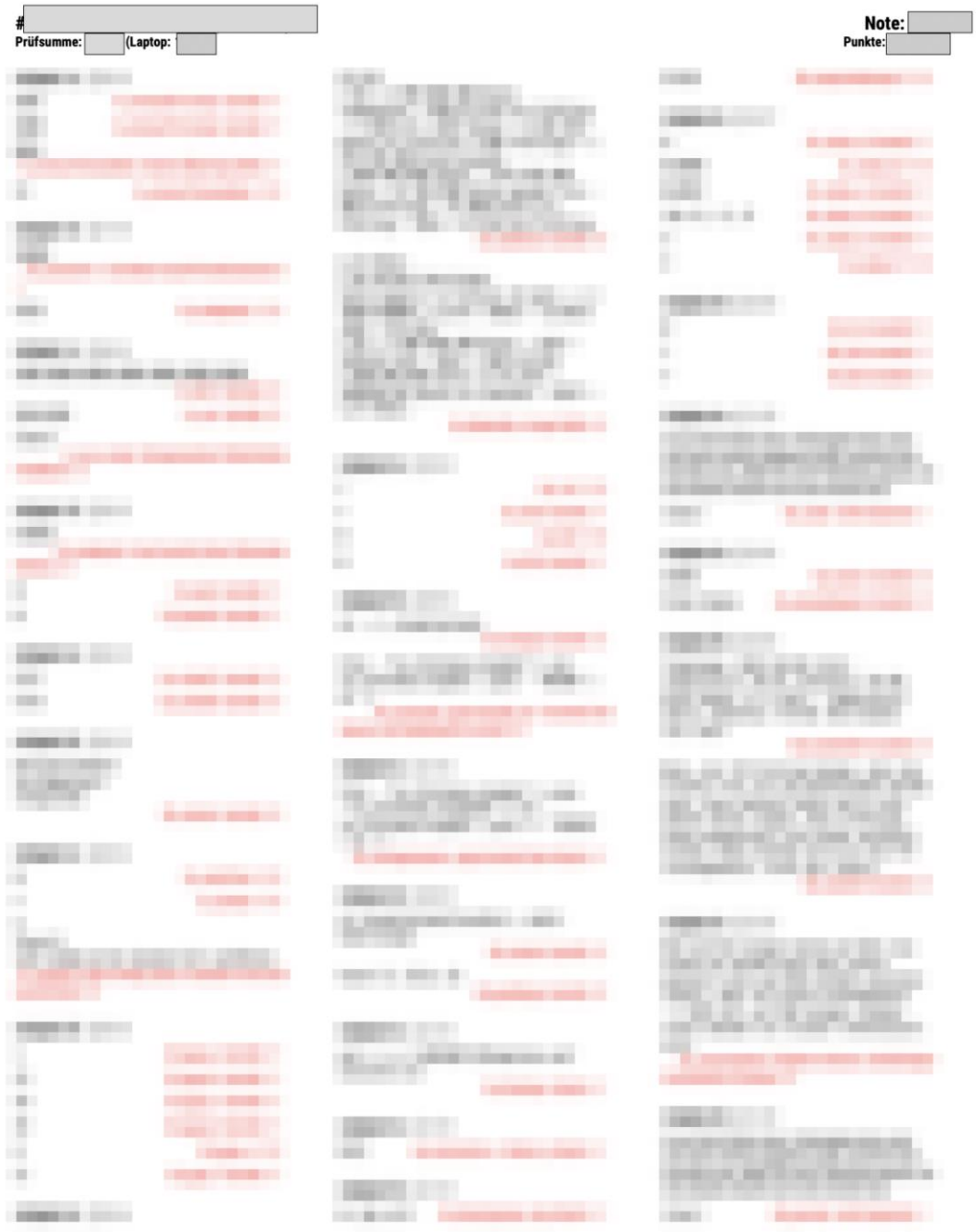
WEITERES VORGEHEN

4 weitere Prüfungen im WiSe

Abstimmung einer Lösung für elektronische **Archivierung**

Nicht gezeigt:

- Digitale Korrektur
- Digitale Einsicht
- Zweitkorrektur



4 weitere Prüfungen im WiSe

Abstimmung einer Lösung für elektronische **Archivierung**

WEITERES VORGEHEN

4 weitere Prüfungen im WiSe

Abstimmung einer Lösung für elektronische **Archivierung**

Was haben wir übersehen?

Nicht gezeigt:

- Digitale Korrektur
- Digitale Einsicht
- Zweitkorrektur

PRAKTISCHE INFORMATIK-KOMPETENZEN INTERAKTIV AN LAPTOPS PRÜFEN

Ein resilientes System für große E-Prüfungen bei kleinen Ressourcen

Prof. Dr. **Dominik Herrmann**

@herdom · herdom.net

LSt Privatsphäre und Sicherheit in Informationssystemen (PSI) an der Universität Bamberg

psi-exam