Chair of Distributed Systems and Security
School of Computation, Information and Technology
Technical University of Munich

Esolution
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Computer Networking and IT Security

Exam:	INHN0012 / Retake	**Date:**	Thursday 6th April, 2023
Examiner:	Prof. Dr.-Ing. Stephan Günther	**Time:**	14:00 – 16:00

	P 1	P 2	P 3	P 4	P 5	P 6
I						
II						

Before we proceed with reading the processing instructions, please answer the following questions. This information helps us to examine learning success depending on participation in individual lecture components. The information is **voluntary** and **not considered for evaluation**, i. e., answers to these questions do not give credits. In order to exclude any influence, this page will not be made accessible during the correction.

a) Did you attend the lecture?

□ 1 (regularly) □ 2 (sometimes) □ 3 (never)

b) Did you attend the tutorials?

□ 1 (regularly) □ 2 (sometimes) □ 3 (never)

Working instructions

- This exam consists of **16 pages** with a total of **6 problems** and the cheatsheet ditributed with the exam. Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 90 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - one **non-programmable pocket calculator**
  - one **analog dictionary** English ↔ native language
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Left room from ____________ to ____________ / Early submission at ____________

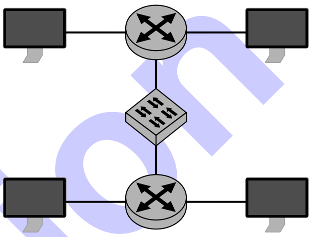
Problem 1 Multiple Choice (14 credits)

The following subproblems are multiple chouce / multiple answer, i. e., at least one answer per subproblem is correct. Sub problems with a single correct answer are graded with 1 credit if correct. Those with more than one correct answers are graded with 0.5 credit per correct answer and −0.5 credit per wrong answer. Missing crosses have no influence. The minimal amount of credits per subproblem is 0 credits.

Mark correct answers with a cross ☒

To undo a cross, completely fill out the answer option ■

To re-mark an option, use a human-readable marking ×■

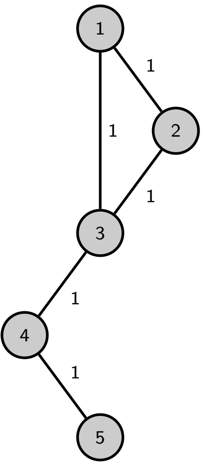


a)* How many broadcast domains does the network to the right contain?

□ 3 ☒ 6 □ 1 □ 5 □ 2 □ 4

b)* How many collision domains does the network to the right contain?

□ 4 □ 2 □ 3 □ 1 ☒ 6 □ 5



c)* Mark the adjacency matrix for the network to the right.

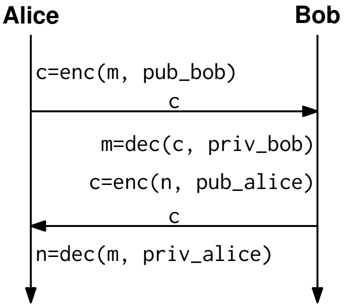
□ $\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$ □ $\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ ☒ $\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

d)* Given the distance matrix $\boldsymbol{D}$ for the network to the right. What is the minimum n such that $\boldsymbol{D}^n = \boldsymbol{D}^{n+1}$ holds?

□ $n = 1$ □ $n = 4$ ☒ $n = 3$ □ $n = 6$

□ $n = 7$ □ $n = 0$ □ $n = 2$ □ $n = 5$

e)* Assume that Bob and Alice know each others public key. What is the scheme on the right vulnerable to?



□ Eve can impersonate Bob

☒ **No** forward-secrecy

□ Man-in-the-Middle and thereby full message decrypt

☒ Replay Attacks

f)* Which **three** of the following claims are true?

□ In AES-CBC, later blocks influence previous ones

□ ECC is robust against quantum computers

□ RSA is robust against quantum computers

☒ AES is robust against quantum computers

□ Common key lengths for AES are 2048 bit and 4096 bit

☒ SHA-256 is vulnerable to length-extension attacks

☒ Cipher text blocks in AES-ECB can be cut and pasted unnoticeable

□ SHA-3 is vulnerable to length-extension attacks g)* What is the AES-CTR scheme?

□ A hash function

□ A block cipher

☒ A stream cipher

□ A key exchange

h)* A chain of trust is used in. . .

□ Trusted fourth parties

☒ Trusted computing

□ Trusted Memory

□ Your trusted bike lock

i)* The domain name system. . .

□ has a mapping from every single IP address to a domain name

☒ translates domain names to IP addresses

□ is inherently trustworthy

□ has a single, central authority

j)* The congestion avoidance phase of TCP Reno. . .

☒ increases the traffic control window linearly

□ is the first phase active in a new connection

□ increases the traffic control window exponentially

□ follows the rapid start phase

k)* In 802.11. . .

☒ management frames are unprotected when using WEP

□ traffic cannot be sniffed by attacker when not in line of sight

□ only two layer 2 addresses are contained in the header

□ arbitrary errors are corrected using the FCS (Frame Correction Sum)

Problem 2 Short Questions: Security (14 credits)

a)* Differentiate Authentication from Authorization.

Authentication is the process of proving an entities identity.
Authorization determines which privileges an entity has.

b) Argue whether an man-in-the-middle attacker can be passive and/or active. Describe a scenario for each applicable property.

A MitM attacker can be both, active and passive.
An attacker eavesdropping on traffic would be passive, while actively replaying traffic puts them into an active position(for explanation).

c) How is a password hash function different from SHA-256?

A password hash function is built to be intentionally slow/expensive to calculate, thereby hindering brute-force attacks.

d)* Name and describe the three properties of a cryptographic hash function.

1. Pre-Image Resistance:
   Given a hash value, it is hard to find an input that results in the same hash.
2. Second Pre-Image Resistance:
   Given a message, it is difficult to find another input that results in the same hash.
3. Collision Resistance:
   It is difficult to find a pair of two different messages that result in the same hash.

e) Briefly describe a scenario in which the IPsec tunnel mode is used.

□ 0 □ 1

A company connects two of its sites (and therefore the two networks) over the public internet.

f) Differentiate a block cipher from a stream cipher.

□ 0 □ 1

A block cipher encrypts entire blocks of data, the plain text has to be padded to a multiple of the block length.
In contrast, a stream cipher generates a key-stream using the underlying block cipher, and can therefore be used to encrypt data byte-wise.

g) Differentiate symmetric encryption from asymmetric encryption. Elaborate on the usage of keys.

□ 0 □ 1 □ 2

Asymmetric encryption uses a key pair, consisting of private and public key, to encrypt and decrypt data. Data that has been encrypted using the public key can only be decrypted using the private key, and vice-versa.
Symmetric encryption is based on a shared secret — the same key is used for encryption and decryption.

h) Describe three functions of a TPM.

□ 0 □ 1 □ 2

- A hardware random number generator
- A way to generate and store keys on the TPM without them ever leaving it
- A way to attest (= identify) a system based on the hardware and software running
- Using the TPM keys to encrypt/decrypt and sign data
- It can be asked to store keys, such as disk encryption keys
- The TPM usually is a separate hardware on the motherboard
- The root key of the TPM is burned into the hardware during production

Problem 3 CRC (14 credits)

In this problem we consider the binary message `00100110` which should be protected by a CRC as we introduced it for Ethernet-based networks in the lecture. We assume the reduction polynomial $r(x) = x^2 + 1$.

a)* Briefly explain what CRC is used for in the context of Ethernet.

Detection of bit errors at the receiving node.

b)* What is the reduction polynomial being used for?

Mapping of a message of arbitrary length to a fixed length checksum.

c)* What does it mean if the reduction polynomial is *irreducible*.

It cannot be represented as the product of two other polynomials of degree strictly less than $\deg r(x)$.

d)* Reason whether or not CRC requires an irreducible reduction polynomial.

It does not: using an irreducible reduction polynomial leads to finite field. However, the purpose of CRC is primarily error detection.Reduible polynomials may have desireable properties such as being able to detect all bit errors of odd length if the factor $(x + 1)$ is contained in the reduction polynomial.

e)* Show whether or not $r(x)$ is irreducible.

$r(x) = x^2 + 1 = (x + 1)^2 \quad \Rightarrow$ it is reducible

f)* Assuming Ethernet, what is the reaction of the receiving node when a bit error is detected.

The frame is dropped without further action.

g)* Determine the CRC checksum for the given message (see beginning of the problem).

```
00100110 00 : 101 = 101101
  101||| ||
  ---||| ||
  00111| ||
    101| ||
    ---| ||
    0100 ||
     101 ||
     --- ||
     001 00
       1 01
       - --
          1
```

Checksum is `01`.

h) Explicitly state the transmitted message.

`00100110 01`

Let us assume a different message (including its checksum): `111011010010111001`. Assume that this message is transmitted and arrives as `111011010010111`**`100`** at the receiver.

i)* Argue whether or not the error is being detected.

It is not detected sind the error is `101`, which is a multiple of the reduction polynomial.

Problem 4 Wireshark (19 credits)

We consider the network topology depicted in Figure 4.1. The PC tries to establish an SSH connection via IPv4 to the server SRV. The MAC and IP addresses of the devices' interfaces are given.

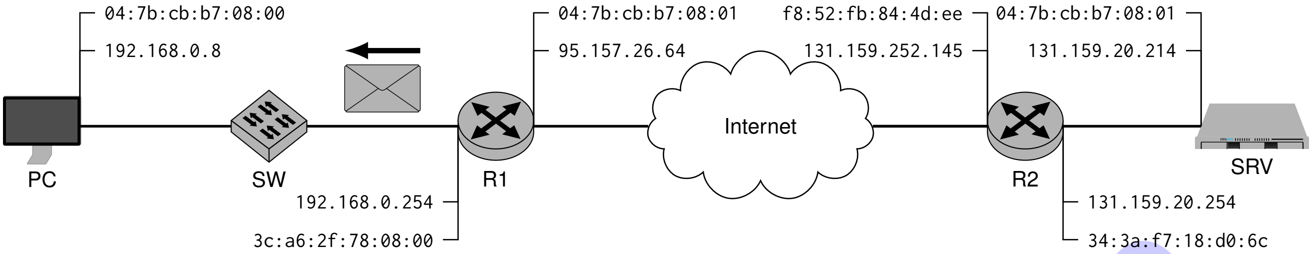
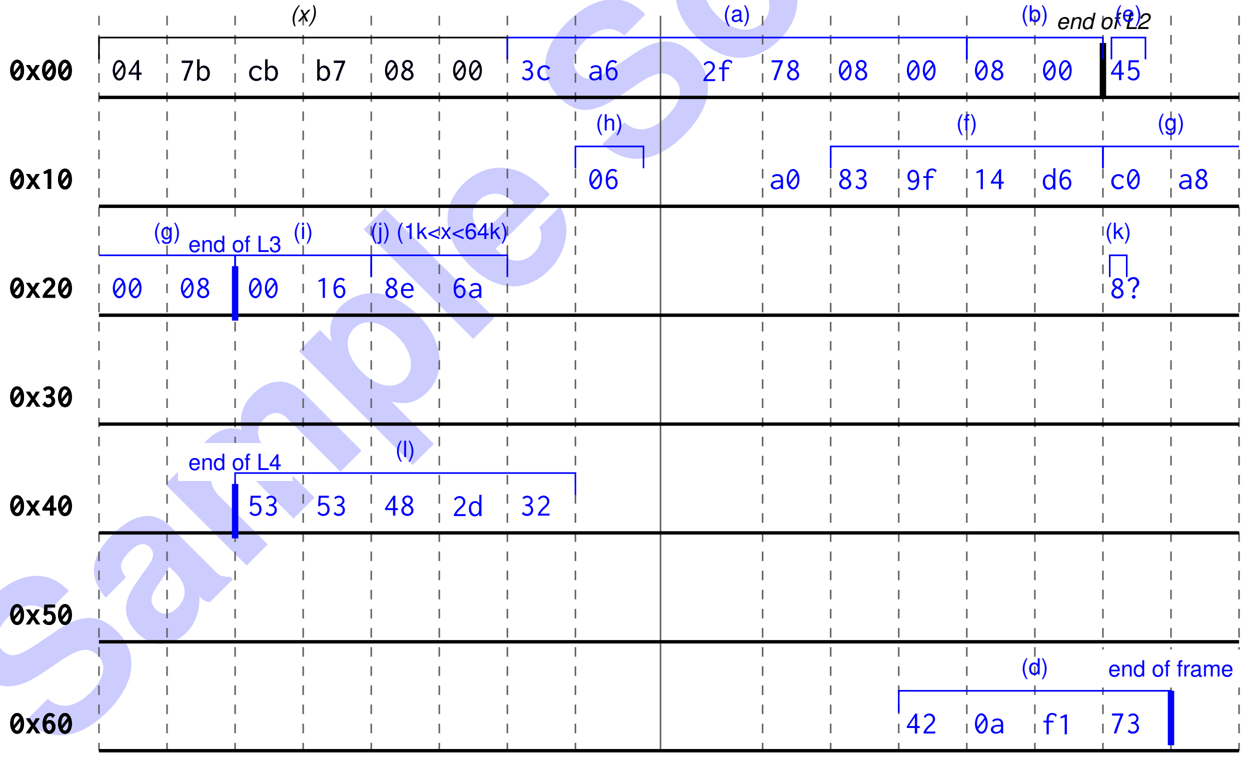


Figure 4.1: Network topology

We consider the frame sent from R1 to the PC as depicted in Figure 4.1, which is the first message from SRV **after** the TCP handshake has completed.

In the following we want to derive the **hexdump of that frame** based on the information given in Figure 4.1 and the following subproblems. Fill in the contents step by step in Figure 4.2. **Make sure to mark to which subproblem your solution belongs**, e. g. by using colors or writing the respective subproblem above your solution. As an example, the L2 receiver address is already filled as answer to some (not existing) Subproblem x).

Notes: There may be some gaps in the final hexdump as we do not derive all contents of that frame. The cheatsheet handed out together with this exam contains any headers and translations you need.



```
0x00  04 7b cb b7 08 00 3c a6  2f 78 08 00 08 00 45
0x10                       06     a0 83 9f 14 d6 c0 a8
0x20  00 08 00 16 8e 6a                            8?
0x30
0x40        53 53 48 2d 32
0x50
0x60                                 42 0a f1 73
```

Figure 4.2: Preprint for the frame's hexdump

0
a)* Fill in the transmitter address of layer 2 in Figure 4.2.

0
1
b)* Fill in the value of the field specifying the type of the L3 PDU in Figure 4.2.

Before we continue to fill out the hexdump, we want to mark the end of different headers. Assume that

- the L3 header does not use any options,
- the L4 header uses 12 B options, and
- the total frame length (including checksum) is 111 B.

c)* Mark **the end** of the **L3** and **L4 headers** as well as of the **frame iteself** in Figure 4.2. As an example, the end of the L2 header is already marked.

d) Fill in the frame check sequence given as `42 0a f1 73` in Figure 4.2.

We now start with filling in different fields of the L3 header. The start of the L3 header is already given in Figure 4.2. **Do not forget to mark to which subproblem your fill ins belong.**

e)* Fill in the field specifying type and length of the L3 header.

f)* Fill in the L3 source address.

g)* Fill in the L3 destination address.

h)* Fill in the value of the field specifying the type of the L3 SDU.

We now continue with filling in different fields of the L4 header. In case a value is not defined, make a reasonable assumption. **Do not forget to mark to which subproblem your fill ins belong.**

i) Fill in the source port.

j) Fill in the destination port.

k) Fill in the value of the field specifying that offset in the L4 header.

Finally, we come the application layer of the frame's content which is the ASCII encoded string "`SSH-2.0-OpenSSH_9.2p1 Debian-2`".

l) Fill in the first 5 B of the L7 PDU.

What is the purpose of this problem?

In this problem students show that they understood

- *how frames are made up of headers of different leayers,*
- *how the beginning/end and type of headers are determined, and*
- *how MAC and IP addresses are used for addressing.*

In addition, they demonstrate that they understood the purpose of MAC and IP addresses as well the basic concept of NAT.

The problem is an exact copy from the CNS Endterm 2023. The only difference is that we do not parse the given hexdump, but create the hexdump from information given in the problem statement.

Given the cheatsheet belonging to this exam, there is no need to remember any specific header layouts.

Problem 5 DNS (13 credits)

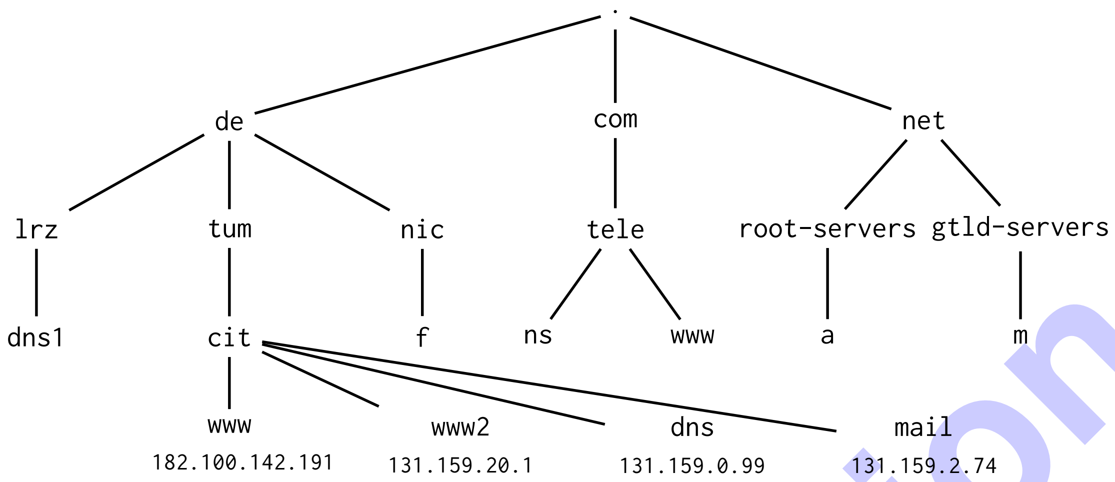


Figure 5.1: A part of the DNS.

a) Briefly describe the purpose of DNS.

The mapping between FQDN and IP addresses.

b) Briefly describe the difference between a fully and non-fully qualified domain name, also regarding their notation.

A fully qualified domain name starts at the root, denoted by the ".".
A non-fully qualified domain name starts at an intermediary node of the DNS.

Figure 5.1 shows the zone file of the authoritative name server for `cit.tum.de`.

```
$ORIGIN cit.tum.de.
$TTL 1H

@ IN SOA dns.cit.tum.de. hostmaster.cit.tum.de. (...)


mail.cit.tum.de.      IN  A    131.159.2.74
www2.cit.tum.de.      IN  A    131.159.20.1
dns.cit.tum.de.       IN  A    131.159.0.99
www.cit.tum.de.       IN  A    182.100.142.191
cit.tum.de.           IN  MX    20 mail.cit.tum.de
cit.tum.de.           IN  NS    dns.cit.tum.de
```

Figure 5.2: DNS zone file on nameserver `dns.cit.tum.de`

c)* Add all other missing data in the zone file depicted in Figure 5.2 based on the information from Figure 5.1.

d)* Distinguish a resolver from a nameserver.

□ 0
□ 1

A resolver is contacted by a client to query the DNS. The resolver then contacts the authoritative nameservers of the zones in order to extract the requested information.

e)* Briefly describe the purpose of a zone's SOA record.

□ 0
□ 1

The SOA (Start of Authority) record defines the root of the zone, as well as the authoritative nameserver.

f)* What does "authoritative" mean in the context of DNS?

□ 0
□ 1

An authoritative nameserver for a zone holds the zone records for this zone and answers queries it.

g)* Determine the PTR record of the address 11.42.43.12. You do not need to reason your answer.

□ 0
□ 1

`12.43.42.11.in-addr.arpa.`

h)* Describe the components of the url `https://www.cit.tum.de/webmail?user=user&pwd=pass`.

□ 0
□ 1
□ 2

1. `https`: the protocol is TLS secured HTTP
2. `www.cit.tum.de`: domain name
3. `/webmail`: path requested on the server
4. `user=user&pwd=pass`: URL parameters / variables, probably a login

i) Explain the difference between recursive and iterative name resolution.

□ 0
□ 1
□ 2

With recursive resolution, only one request for a resource record is made to a configured resolver, which returns the final response.
With iterative resolution, the FQDN is instead resolved starting at the root zone (or the last known SOA) by querying the authoritative name servers for the respective zones.Their answers contain either the FQDN of an authoritative name server of the next lower zone or the final resource record if the queried name server is authoritative for it.

Problem 6 Side-Channel Information Stealing (16 credits)

You have breached the data center of a large cloud hosting provider. You intend to extract their private key of 4096 bit length. The key is derived from perfect, uniform randomness. As a very strict network policy is employed there is no way you will be able to do this via the network.
Therefore, you have come up with another way: The cloud hosting provider streams their data center via a live cam (24 frames per second), seemingly to show off their hardware. This stream includes a view of the hard drive activity LED of the relevant server. You can control the LED — and decide to extract the private key by encoding it through LED blink patterns, which you can decode by viewing the live stream.

0 □ 1 □ 2 □

a)* What is the maximum data rate achievable on the channel?

The LED has two states: on and off. Therefore, each frame shows one of two symbols. Each state therefore encodes $\log_2 2 = 1$ bit.
We can detect a different state with each new frame, therefore the maximum channel rate is 24 symbols per second.
This results in a maximum data rate of $r = 24\,\text{bit/s}$.

To properly detect the LED being on or off, a **pulse length of at least** 100 ms is needed.

0 □ 1 □ 2 □

b) What is the resulting transmission rate? How long does it take to transmit the private key?

We can transmit $\frac{1\,\text{s}}{100\,\text{ms}} = 10$ symbols per second, therefore have a transmission rate of 10 bit/s.
It takes $4096/10 \approx 409.6\,\text{s}$ to transmit the key.

There is still one problem remaining: you have to properly synchronize the transmission on sender and receiver side. To achieve this you decide to employ a new *8b11b* coding scheme. In this scheme, 8 bit of payload are coded to 11 bit of channel word. The coding transforms a byte to a channel word by prepending a start sequence, thereby marking the start of each channel word recognizeable. This start sequence is **three bits** long, and consists of all ones: 111.
To not confuse frame starts with actual data, all left-most occurrences of 11 in the data are replaced with `110`. This process is called bit-stuffing. On the receiving side, this process is reversed, thus replacing `110` by 11.
You can neglect any padding that would become necessary for all following sub tasks!

0 □ 1 □ 2 □ 3 □

c)* Determine the expected length **increase** of the actual transmitted data. Note, that the bit-stuffing is done on the payload!

Let $k \in \{0,1\}^{4096}$ be uniformly random.

We define $X_i = \begin{cases} 1 & s_i s_{i+1} = 11 \quad \text{for } 1 \le i \le n-1 \\ 0 & \textit{else} \end{cases}$

The length increase is equal to $X = \sum_{i=1}^{n-1} X_i$

Thereby, $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^{n-1} X_i] = \sum_{i=1}^{n-1} \mathbb{E}[X_i] = \sum_{i=1}^{n-1} \mathbb{P}[X_i = 1] = \sum_{i=1}^{n-1} \frac{1}{4} = \frac{n-1}{4} = 1023.75$

If you were unable to solve subproblem c), use 1024 bit **as the expected length increase.**

d) Using the expected length increase, determine the expected code rate.

□ 0 □ 1 □ 2

A frame start sequence of 3 bit is added for every octett.
Additionally, we have to stuff the 11 bit sequences, as determined prior.
Therefore:
Expected length: $L = 4096 + \lfloor \frac{4096}{8} \rfloor \cdot 3 + 1023.75 = 6655.75\,\text{bit}$
And by that:
$R = \frac{4096}{6655.75} \approx 0.62$

e)* Argue whether it is realistic to calculate the key by brute-force, rather than extracting it via the side-channel.

□ 0 □ 1

No, brute-forcing 4096 bit is not realistically doable, even given multiple years of time.

You decide that your current approach is too inefficient.

f) Propose an approach to reducing the overhead while maintaining the synchronization properties.

□ 0 □ 1

E. g. instead of using the 8b11b coding, we can use a 256b259b coding to synchronize sender and receiver. By doing so we reduce the overhead.

While testing your approach you realize that there is still a $\frac{1}{200}$ chance left that you read a given bit **incorrectly**. Given this, you decide to add redundancy to your coding, extending the *8b11b* coding to the **8b11b_v2** coding. Again, each 8 bit long word of data is translated to 11 bit of channel word. In contrast to before, the coding scheme employed, which shall not be explained in detail, allows for synchronization **without** the use of special symbols. Bit-stuffing is therefore not longer necessary.

0 □□ 1 □□ 2 □□

g) How long is the resulting data sent? Is this coding on average more efficient than the previous approach?

We have $\frac{4096}{8} = 512$ data words, which are translated to 512 channel words, each 11 bit long.
The total (constant) length therefore is $L_{new} = \frac{4096}{8} \cdot 11\,\text{bit} = 5632\,\text{bit}$
Therefore this approach is more efficient than the previous approach ($5632\,\text{bit} < 6655.75\,\text{bit}$).This is as we no longer require bit-stuffing.

The new coding additionally allows you to **detect and correct** one flipped bit in each **channel word**.

0 □□ 1 □□ 2 □□ 3 □□

h) What is the probability Pr[incorrect] that the transmission cannot be decoded correctly?
Note: For this sub task, **round to four digits of accuracy**. You are allowed to calculate using rounded interim results

Let X_i be the number of flipped bits in channel word i.
We can recover the word correctly if zero or one bits flip. If two or more bits flip, the transmission is decoded incorrectly.
We continue with the channel words individually:
$\Pr[X_i = 0] = (1 - \frac{1}{200})^{11} \approx 0.9464$
$\Pr[X_i = 1] = (1 - \frac{1}{200})^{10} \cdot (\frac{1}{200})^{1} \cdot \binom{11}{1} \approx 0.0523$
Therefore $\Pr[\text{correct}] = \Pr[X_i = 0] + \Pr[X_i = 1] \approx 0.9987$
We cannot decode the data if one or more of the 512 channel words fail in decoding.
Therefore $\Pr[\text{incorrect}] = 1 - (\Pr[\text{correct}])^{512} \approx 1 - 0.5137 = 0.4863$
Note: without interim result rounding, the result is 0.4952!

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.