Prompting Steganography: A New Paradigm

Hanzhou Wu

Shanghai University

Jan. 23, 2024

Outline

1 Introduction

2 Prompting Steganography

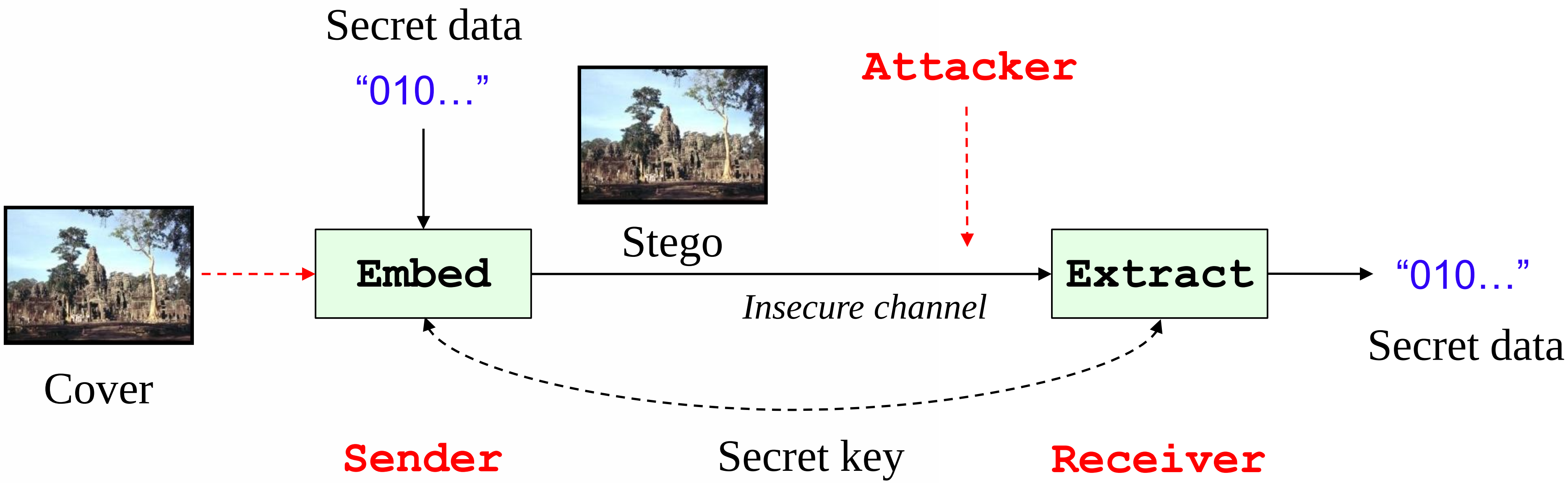
3 Prompt Optimization

4 Conclusion and Discussion

1 Introduction

❑ Steganography

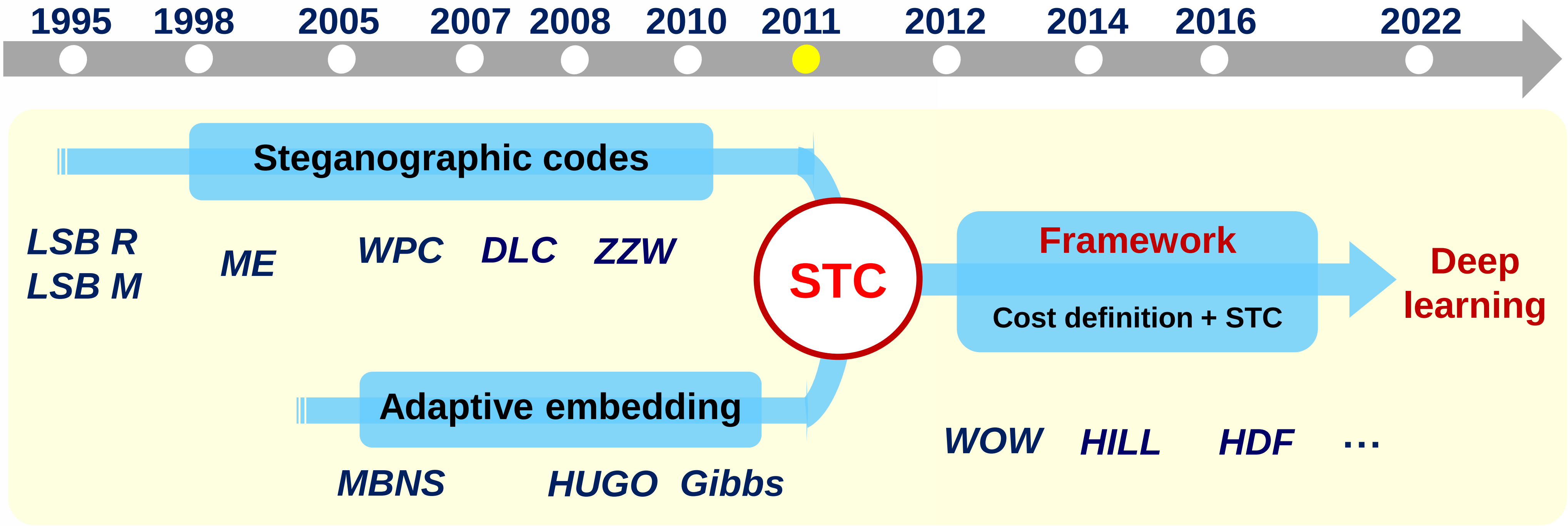
- ❑ Hides secret data into a digital covert for covert communication
- ❑ Conceals the existence of the present communication



1 Introduction

History of Steganography

- Early methods: modify cover elements as few as possible
- Now: modify low-cost cover elements, or apply deep learning



1995 1998 2005 2007 2008 2010 2011 2012 2014 2016 2022

Steganographic codes

LSB R
LSB M

ME

WPC *DLC* *ZZW*

STC

Framework

Cost definition + STC

Deep learning

Adaptive embedding

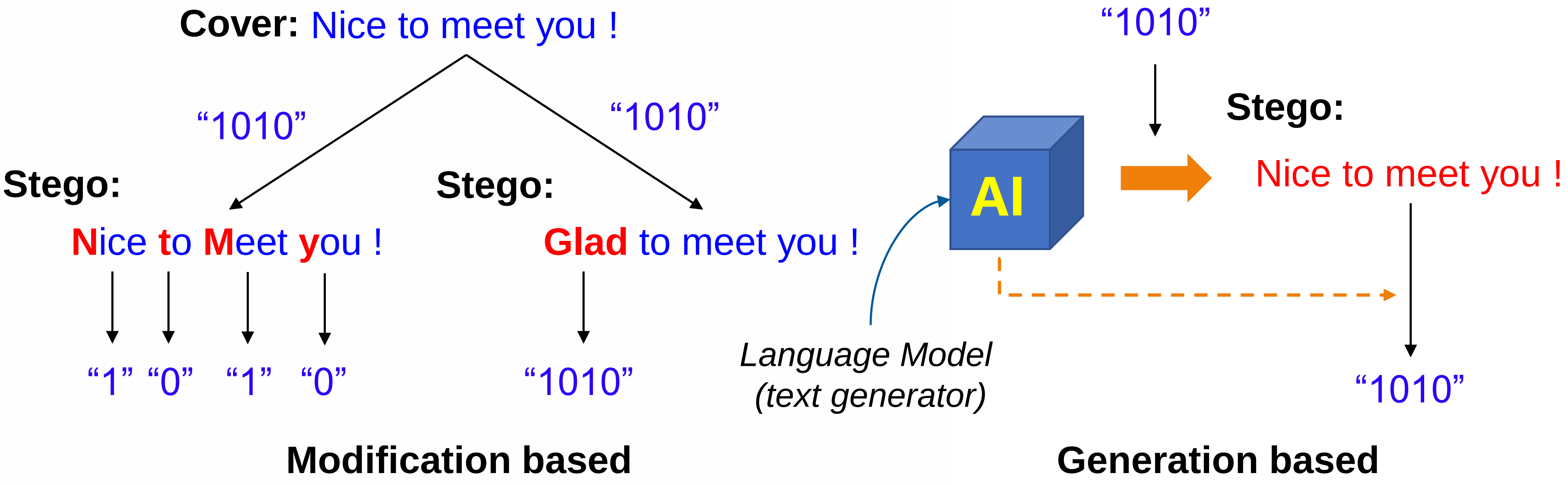
MBNS *HUGO* *Gibbs*

WOW *HILL* *HDF* …

1 Introduction

History of Steganography

- Mainstream methods: modification based (requiring a cover)
- Recently: generation based (without a cover)

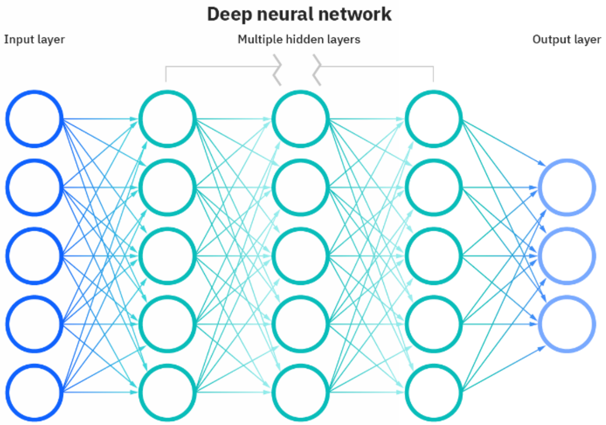


1 Introduction

Thoughts and Motivation

- On one hand: we can continue to improve the existing methods
- On the other hand: what will be the next generation of steganography?

First generation:
Hand-crafted



Second generation:
Learning based

Next generation:
Reasoning based ?

Outline

1. Introduction
2. Prompting Steganography
3. Prompt Optimization
4. Conclusion and Discussion

2 Prompting Steganography

☐ Language model (LM)

- ☐ A probabilistic model that uses machine learning to conduct a probability distribution over sequences of tokens (or say words)
- ☐ Learns from textual data and has various applications such as text generation, text classification, and language translation
- ☐ Early LMs are built upon statistical approaches such as Markov process and Bayesian analysis
- ☐ Recent LMs are based on RNN, LSTM and Transformer

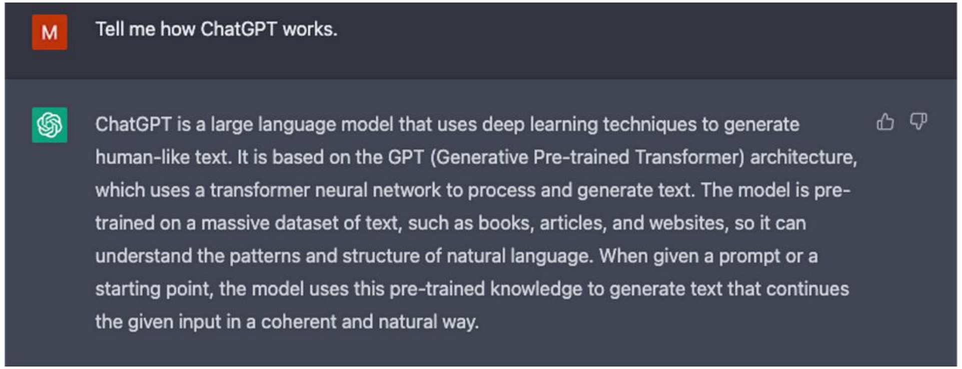
2 Prompting Steganography

☐ Large language model (LLM)

- ☐ Language models are trained with a huge number of texts in advance so that they are also called *pre-trained language models (PLMs)*
- ☐ PLMs are being developed along the direction that large language models (LLMs) show better performance on downstream tasks
- ☐ LLM is characterized by its large size, e.g., LaMDA has around 137 billion parameters, GPT-3 has around 175 billion parameters, Gopher has around 280 billion parameters and so on

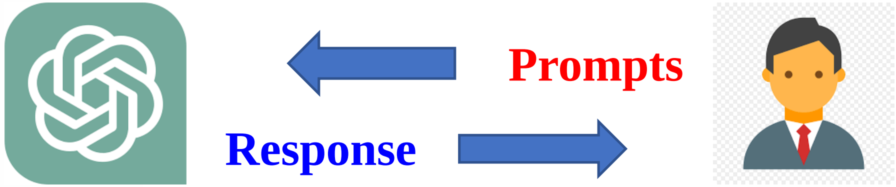
2 Prompting Steganography

- **Large language model (LLM)**
  - LLMs have strong ability to understand natural language and solve complex problems by text generation
  - ***ChatGPT:*** *a powerful LLM capable of generating human-like text based on context and past conversations*



M Tell me how ChatGPT works.

ChatGPT is a large language model that uses deep learning techniques to generate human-like text. It is based on the GPT (Generative Pre-trained Transformer) architecture, which uses a transformer neural network to process and generate text. The model is pre-trained on a massive dataset of text, such as books, articles, and websites, so it can understand the patterns and structure of natural language. When given a prompt or a starting point, the model uses this pre-trained knowledge to generate text that continues the given input in a coherent and natural way.



2 Prompting Steganography

- **Large language model (LLM)**
  - **Emergent abilities** of LLMs are defined as those abilities that are not present in smaller models but are present in larger models
  - **Typical emergent abilities:** *in-context learning*, *instruction following and step-by-step reasoning*
  - **In-context learning** enables LLMs to infer how to perform a new downstream task from a few examples in the context without training
  - **Instruction following** means LLMs can follow the instructions for new tasks without using explicit examples
  - **Step-by-step reasoning** allows LLMs to solve many complex tasks such as math and reasoning problems

2 Prompting Steganography

❑ Prompt Engineering

- ❑ LLMs have strong reasoning ability to solve complex tasks
- ❑ This reasoning ability can be enhanced by **prompt engineering**
- ❑ **Prompt engineering:** to develop and optimize prompts for LLMs so that LLMs can return the better solution

➢ **Input:** 100 + 200 * 300 = ?

➢ **Output:** The answer is **90000**. ✗

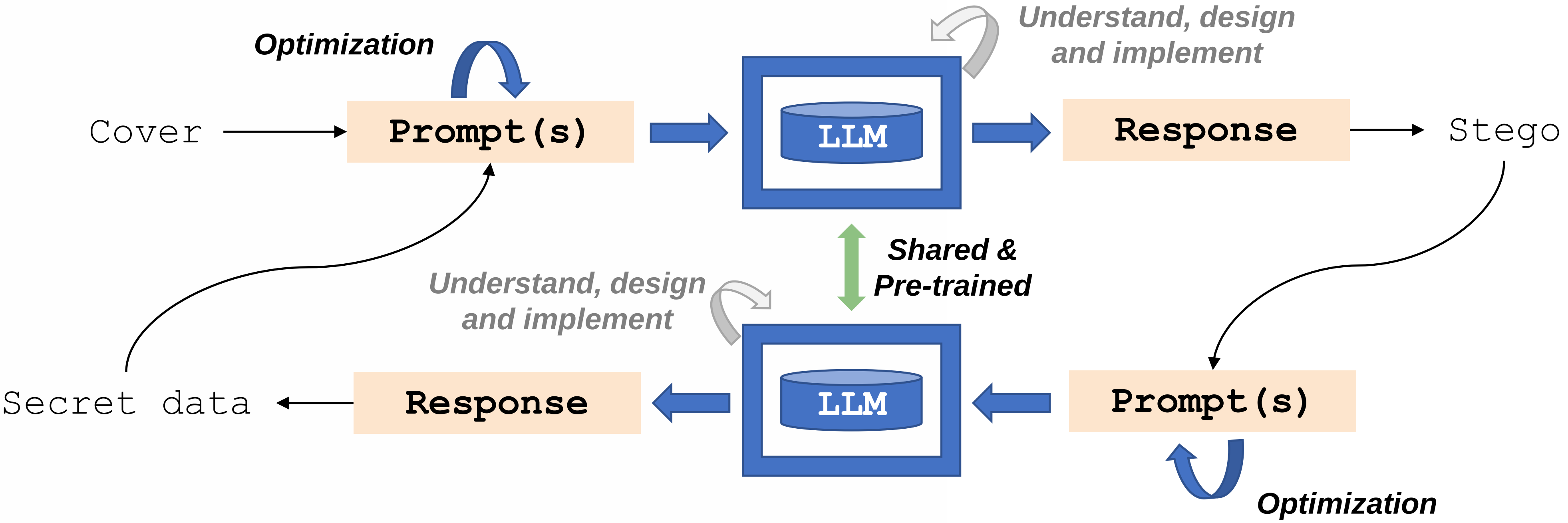
➢ **Input:** 100 + 200 * 300 = ? Let’s think step by step.

➢ **Output:** First, we have 200 * 300 = 60000. Then, we determine 100 + 60000 = 60100. So, the result is **60100**. ✓

2 Prompting Steganography

❑ Prompting Steganography

- ❑ Through reasoning, an LLM can realize steganography by itself

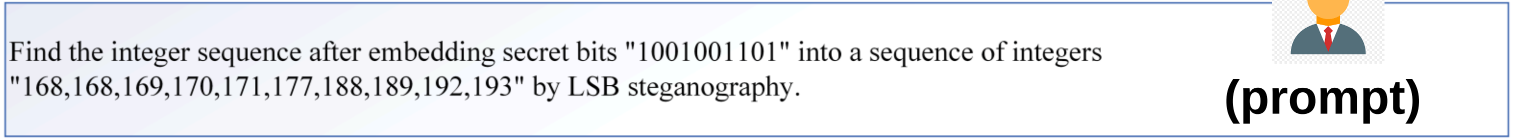


2 Prompting Steganography

- **Prompting Steganography**
  - **Zero-shot steganography**
    - Assigns steganography tasks directly to LLMs without demonstrations/examples
  - **Few-shot steganography**
    - Provides a few of steganography demonstrations/examples to the LLM
    - So that the LLM can better understand and finish the steganography mission
    - One-shot steganography: the number of demonstrations/examples equals 1
  - Examples: next pages

2 Prompting Steganography

LLM input:

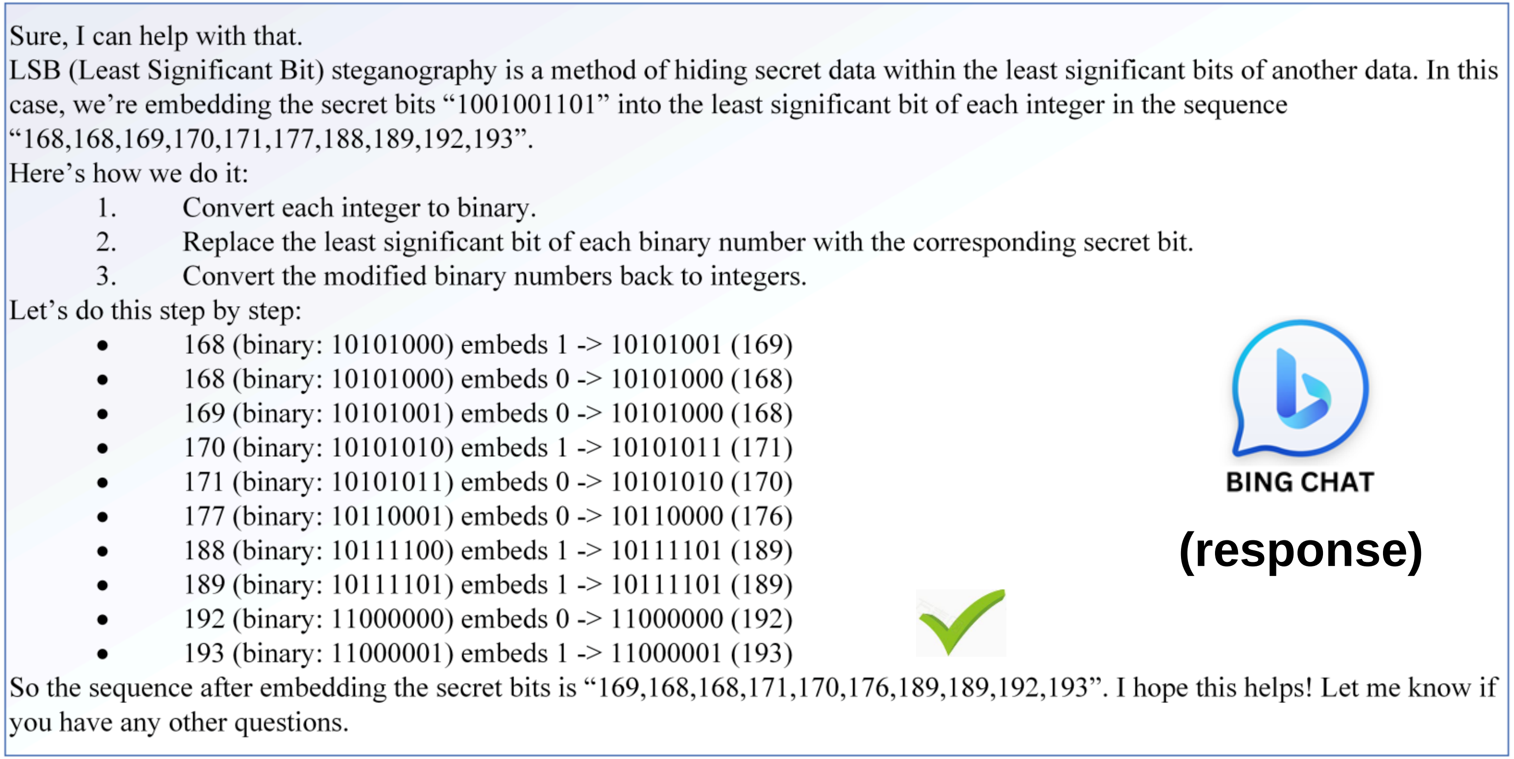


Find the integer sequence after embedding secret bits "1001001101" into a sequence of integers "168,168,169,170,171,177,188,189,192,193" by LSB steganography.

(prompt)

Zero-shot scenario

LLM output:



Sure, I can help with that.

LSB (Least Significant Bit) steganography is a method of hiding secret data within the least significant bits of another data. In this case, we're embedding the secret bits "1001001101" into the least significant bit of each integer in the sequence "168,168,169,170,171,177,188,189,192,193".

Here's how we do it:

1. Convert each integer to binary.
2. Replace the least significant bit of each binary number with the corresponding secret bit.
3. Convert the modified binary numbers back to integers.

Let's do this step by step:

- 168 (binary: 10101000) embeds 1 -> 10101001 (169)
- 168 (binary: 10101000) embeds 0 -> 10101000 (168)
- 169 (binary: 10101001) embeds 0 -> 10101000 (168)
- 170 (binary: 10101010) embeds 1 -> 10101011 (171)
- 171 (binary: 10101011) embeds 0 -> 10101010 (170)
- 177 (binary: 10110001) embeds 0 -> 10110000 (176)
- 188 (binary: 10111100) embeds 1 -> 10111101 (189)
- 189 (binary: 10111101) embeds 1 -> 10111101 (189)
- 192 (binary: 11000000) embeds 0 -> 11000000 (192)
- 193 (binary: 11000001) embeds 1 -> 11000001 (193)

So the sequence after embedding the secret bits is "169,168,168,171,170,176,189,189,192,193". I hope this helps! Let me know if you have any other questions.

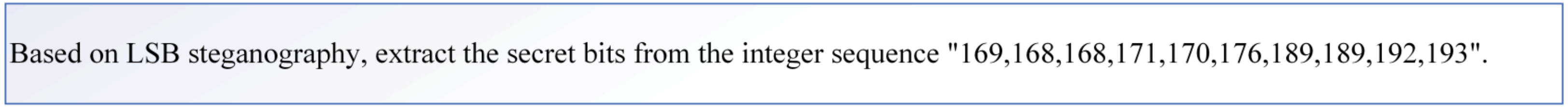
BING CHAT

(response)

data embedding

2 Prompting Steganography

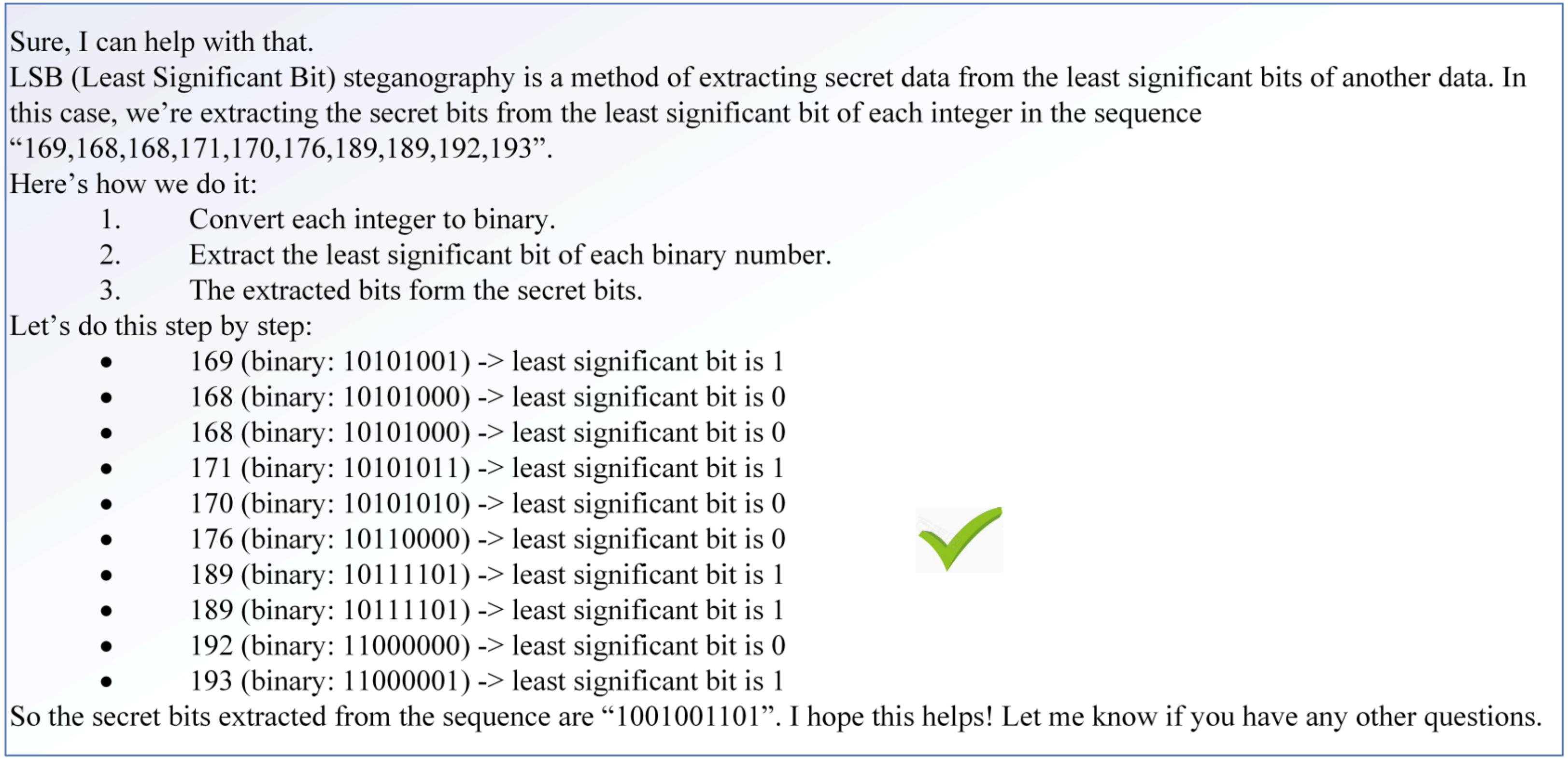
LLM input:



Based on LSB steganography, extract the secret bits from the integer sequence "169,168,168,171,170,176,189,189,192,193".

Zero-shot scenario

LLM output:



Sure, I can help with that.

LSB (Least Significant Bit) steganography is a method of extracting secret data from the least significant bits of another data. In this case, we're extracting the secret bits from the least significant bit of each integer in the sequence "169,168,168,171,170,176,189,189,192,193".

Here's how we do it:

1. Convert each integer to binary.
2. Extract the least significant bit of each binary number.
3. The extracted bits form the secret bits.

Let's do this step by step:

- 169 (binary: 10101001) -> least significant bit is 1
- 168 (binary: 10101000) -> least significant bit is 0
- 168 (binary: 10101000) -> least significant bit is 0
- 171 (binary: 10101011) -> least significant bit is 1
- 170 (binary: 10101010) -> least significant bit is 0
- 176 (binary: 10110000) -> least significant bit is 0
- 189 (binary: 10111101) -> least significant bit is 1
- 189 (binary: 10111101) -> least significant bit is 1
- 192 (binary: 11000000) -> least significant bit is 0
- 193 (binary: 11000001) -> least significant bit is 1

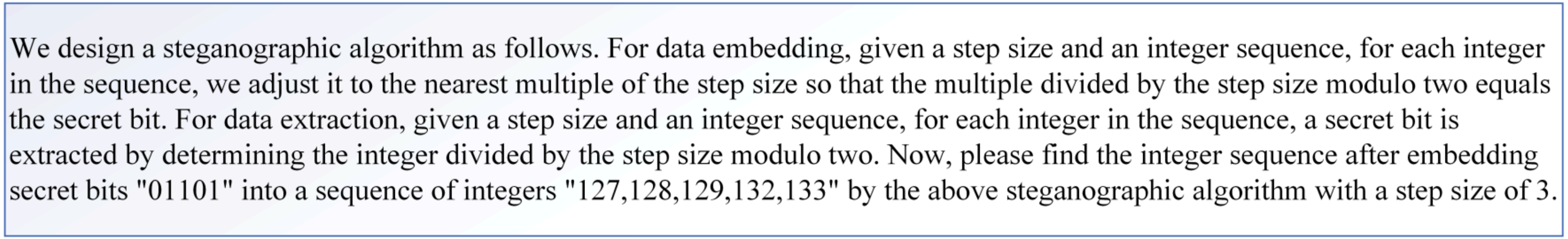
So the secret bits extracted from the sequence are "1001001101". I hope this helps! Let me know if you have any other questions.

✓

data extraction

2 Prompting Steganography

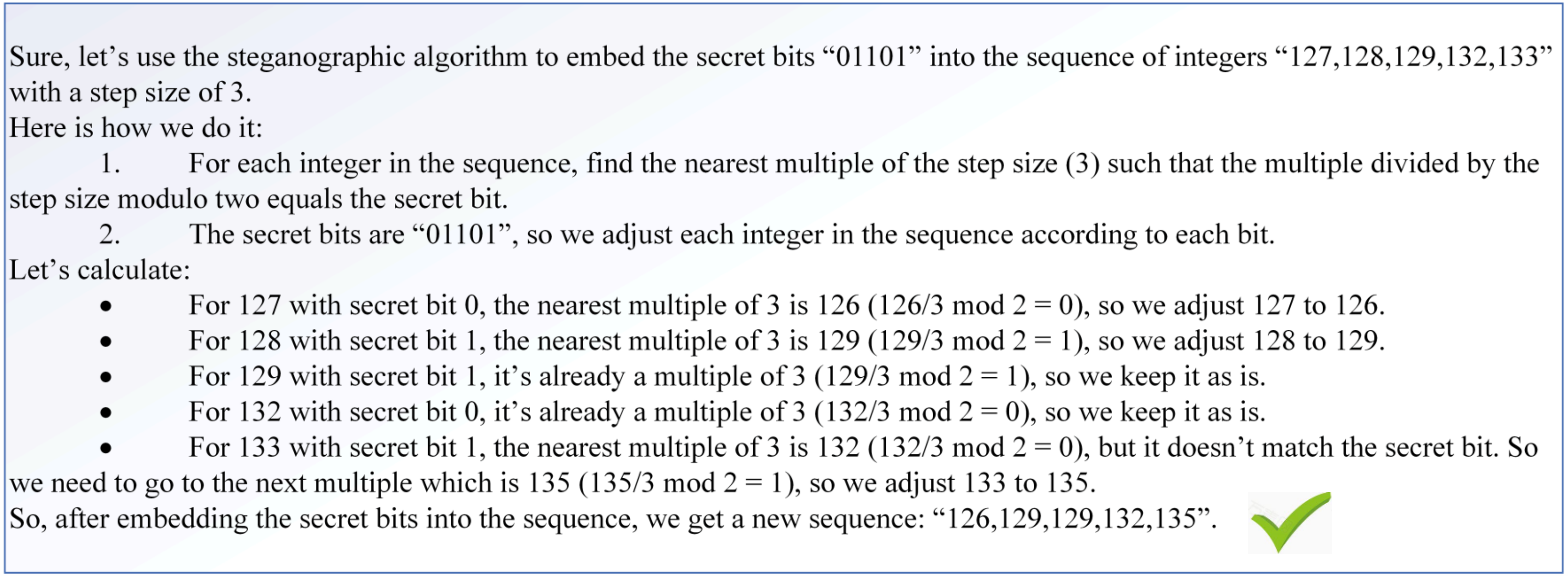
LLM input:



We design a steganographic algorithm as follows. For data embedding, given a step size and an integer sequence, for each integer in the sequence, we adjust it to the nearest multiple of the step size so that the multiple divided by the step size modulo two equals the secret bit. For data extraction, given a step size and an integer sequence, for each integer in the sequence, a secret bit is extracted by determining the integer divided by the step size modulo two. Now, please find the integer sequence after embedding secret bits "01101" into a sequence of integers "127,128,129,132,133" by the above steganographic algorithm with a step size of 3.

Zero-shot scenario

LLM output:



Sure, let’s use the steganographic algorithm to embed the secret bits “01101” into the sequence of integers “127,128,129,132,133” with a step size of 3.

Here is how we do it:

1. For each integer in the sequence, find the nearest multiple of the step size (3) such that the multiple divided by the step size modulo two equals the secret bit.
2. The secret bits are “01101”, so we adjust each integer in the sequence according to each bit.

Let’s calculate:

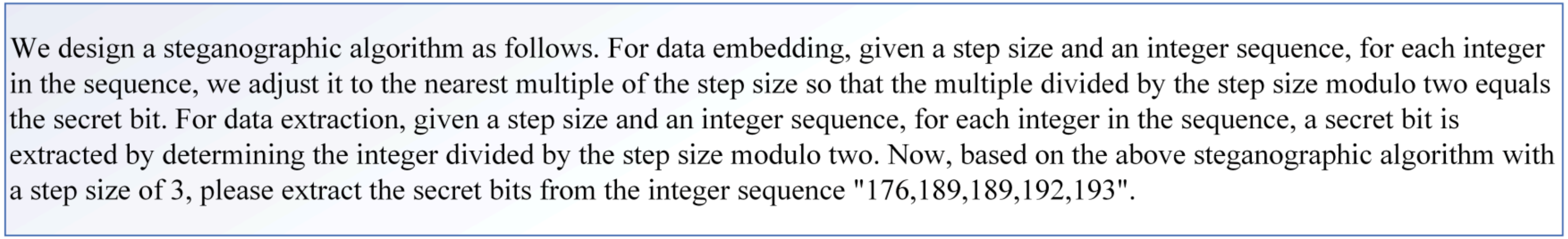
- For 127 with secret bit 0, the nearest multiple of 3 is 126 (126/3 mod 2 = 0), so we adjust 127 to 126.
- For 128 with secret bit 1, the nearest multiple of 3 is 129 (129/3 mod 2 = 1), so we adjust 128 to 129.
- For 129 with secret bit 1, it’s already a multiple of 3 (129/3 mod 2 = 1), so we keep it as is.
- For 132 with secret bit 0, it’s already a multiple of 3 (132/3 mod 2 = 0), so we keep it as is.
- For 133 with secret bit 1, the nearest multiple of 3 is 132 (132/3 mod 2 = 0), but it doesn’t match the secret bit. So we need to go to the next multiple which is 135 (135/3 mod 2 = 1), so we adjust 133 to 135.

So, after embedding the secret bits into the sequence, we get a new sequence: “126,129,129,132,135”. ✓

data embedding

2 Prompting Steganography

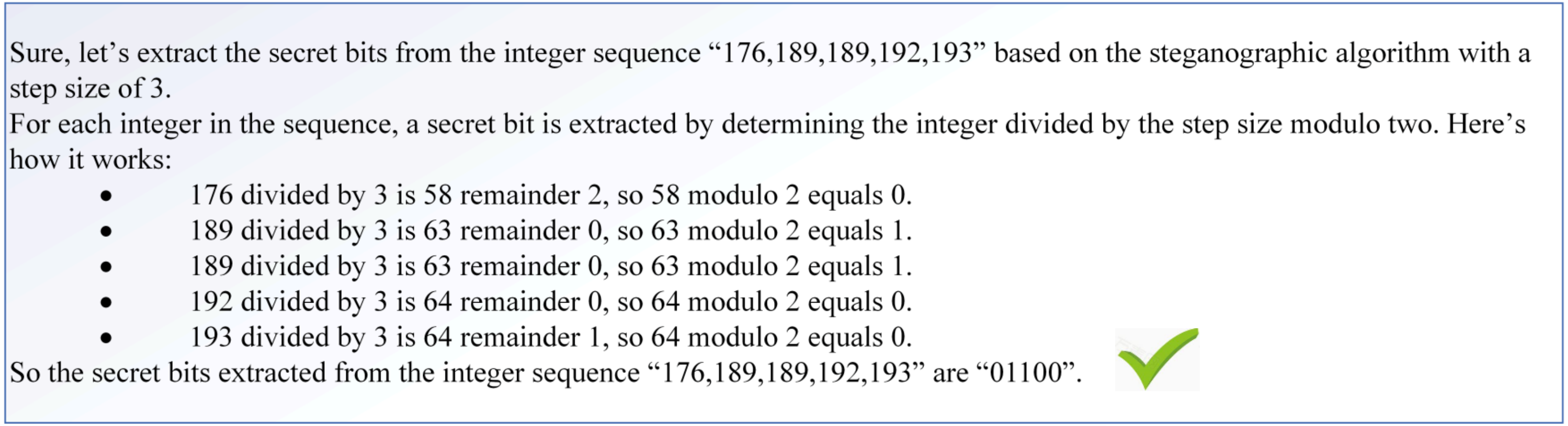
LLM input:



We design a steganographic algorithm as follows. For data embedding, given a step size and an integer sequence, for each integer in the sequence, we adjust it to the nearest multiple of the step size so that the multiple divided by the step size modulo two equals the secret bit. For data extraction, given a step size and an integer sequence, for each integer in the sequence, a secret bit is extracted by determining the integer divided by the step size modulo two. Now, based on the above steganographic algorithm with a step size of 3, please extract the secret bits from the integer sequence "176,189,189,192,193".

Zero-shot scenario

LLM output:



Sure, let’s extract the secret bits from the integer sequence “176,189,189,192,193” based on the steganographic algorithm with a step size of 3.

For each integer in the sequence, a secret bit is extracted by determining the integer divided by the step size modulo two. Here’s how it works:

- 176 divided by 3 is 58 remainder 2, so 58 modulo 2 equals 0.
- 189 divided by 3 is 63 remainder 0, so 63 modulo 2 equals 1.
- 189 divided by 3 is 63 remainder 0, so 63 modulo 2 equals 1.
- 192 divided by 3 is 64 remainder 0, so 64 modulo 2 equals 0.
- 193 divided by 3 is 64 remainder 1, so 64 modulo 2 equals 0.

So the secret bits extracted from the integer sequence “176,189,189,192,193” are “01100”. ✓

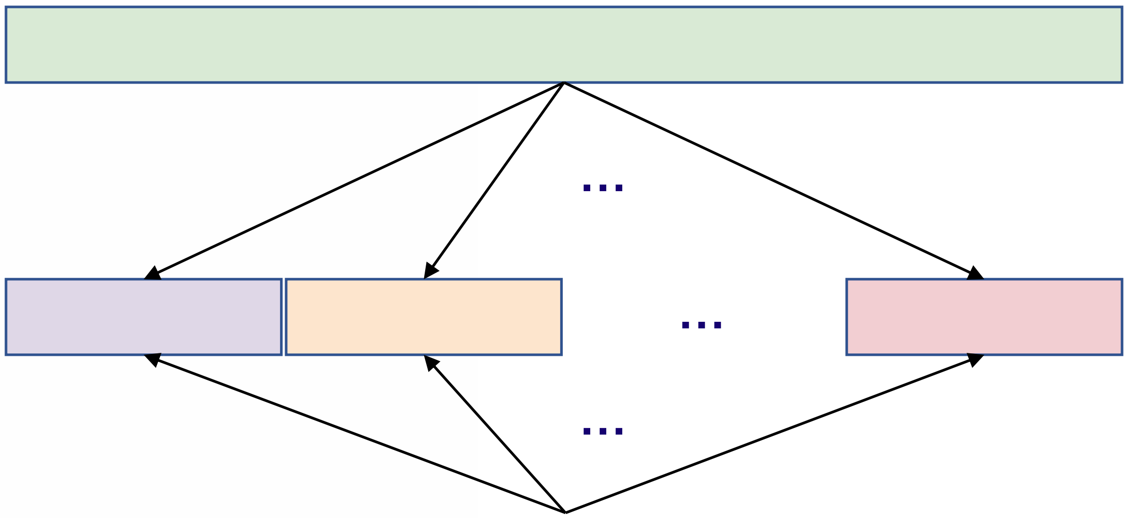
data extraction

2 Prompting Steganography

(prompt)

Please design a new steganographic algorithm that segments a cover sequence into multiple subsequences and uses each subsequence to carry two bits. The length of each subsequence should be larger than 1.

(response)



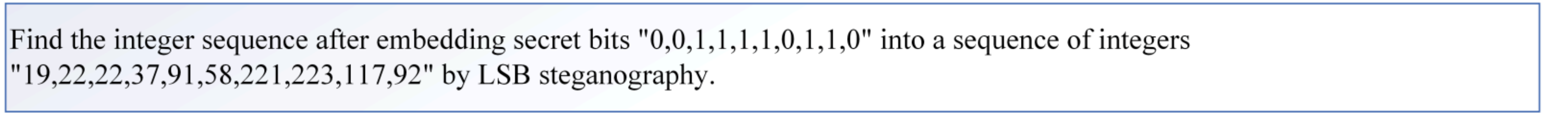
Original cover

Sub-covers

LSB embedding: the last two elements

2 Prompting Steganography

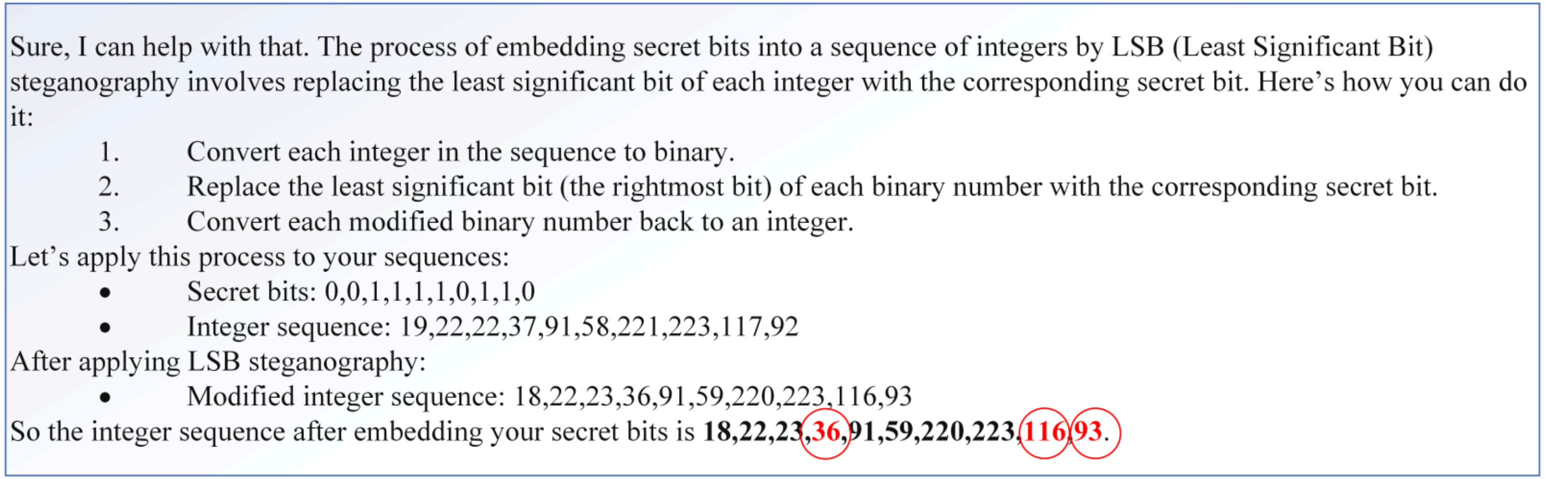
LLM input:



Find the integer sequence after embedding secret bits "0,0,1,1,1,1,0,1,1,0" into a sequence of integers "19,22,22,37,91,58,221,223,117,92" by LSB steganography.

Zero-shot scenario

LLM output:



Sure, I can help with that. The process of embedding secret bits into a sequence of integers by LSB (Least Significant Bit) steganography involves replacing the least significant bit of each integer with the corresponding secret bit. Here's how you can do it:

1. Convert each integer in the sequence to binary.
2. Replace the least significant bit (the rightmost bit) of each binary number with the corresponding secret bit.
3. Convert each modified binary number back to an integer.

Let's apply this process to your sequences:

- Secret bits: 0,0,1,1,1,1,0,1,1,0
- Integer sequence: 19,22,22,37,91,58,221,223,117,92

After applying LSB steganography:

- Modified integer sequence: 18,22,23,36,91,59,220,223,116,93

So the integer sequence after embedding your secret bits is **18,22,23,36,91,59,220,223,116,93**.

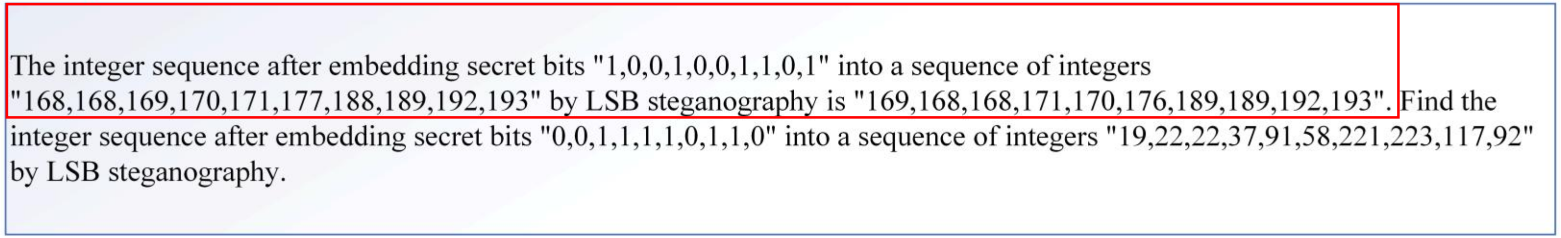
data embedding

✗

2 Prompting Steganography

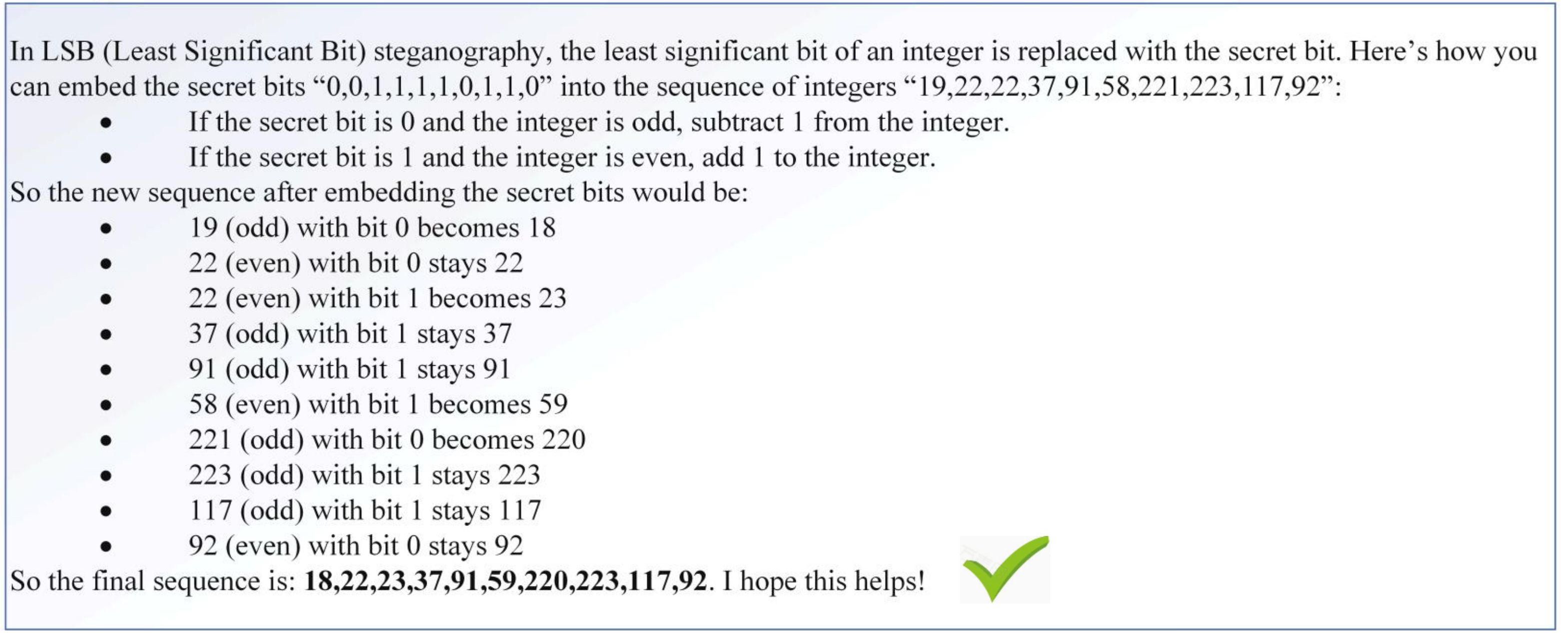
LLM input:

examples


The integer sequence after embedding secret bits "1,0,0,1,0,0,1,1,0,1" into a sequence of integers "168,168,169,170,171,177,188,189,192,193" by LSB steganography is "169,168,168,171,170,176,189,189,192,193". Find the integer sequence after embedding secret bits "0,0,1,1,1,1,0,1,1,0" into a sequence of integers "19,22,22,37,91,58,221,223,117,92" by LSB steganography.

Few-shot scenario

LLM output:


In LSB (Least Significant Bit) steganography, the least significant bit of an integer is replaced with the secret bit. Here's how you can embed the secret bits "0,0,1,1,1,1,0,1,1,0" into the sequence of integers "19,22,22,37,91,58,221,223,117,92":

- If the secret bit is 0 and the integer is odd, subtract 1 from the integer.
- If the secret bit is 1 and the integer is even, add 1 to the integer.

So the new sequence after embedding the secret bits would be:

- 19 (odd) with bit 0 becomes 18
- 22 (even) with bit 0 stays 22
- 22 (even) with bit 1 becomes 23
- 37 (odd) with bit 1 stays 37
- 91 (odd) with bit 1 stays 91
- 58 (even) with bit 1 becomes 59
- 221 (odd) with bit 0 becomes 220
- 223 (odd) with bit 1 stays 223
- 117 (odd) with bit 1 stays 117
- 92 (even) with bit 0 stays 92

So the final sequence is: **18,22,23,37,91,59,220,223,117,92**. I hope this helps! ✓

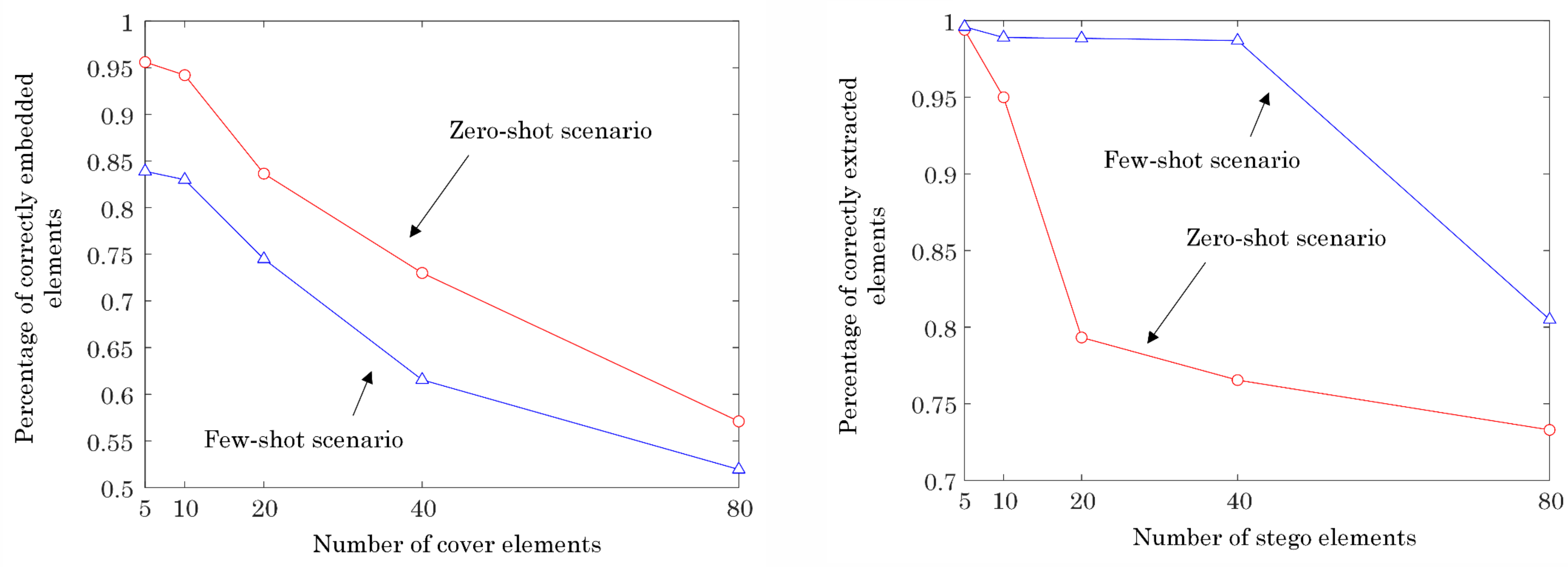
data embedding

2 Prompting Steganography

Prompting Steganography

- Few-shot steganography: always better than zero-shot steganography?
  - **NO**

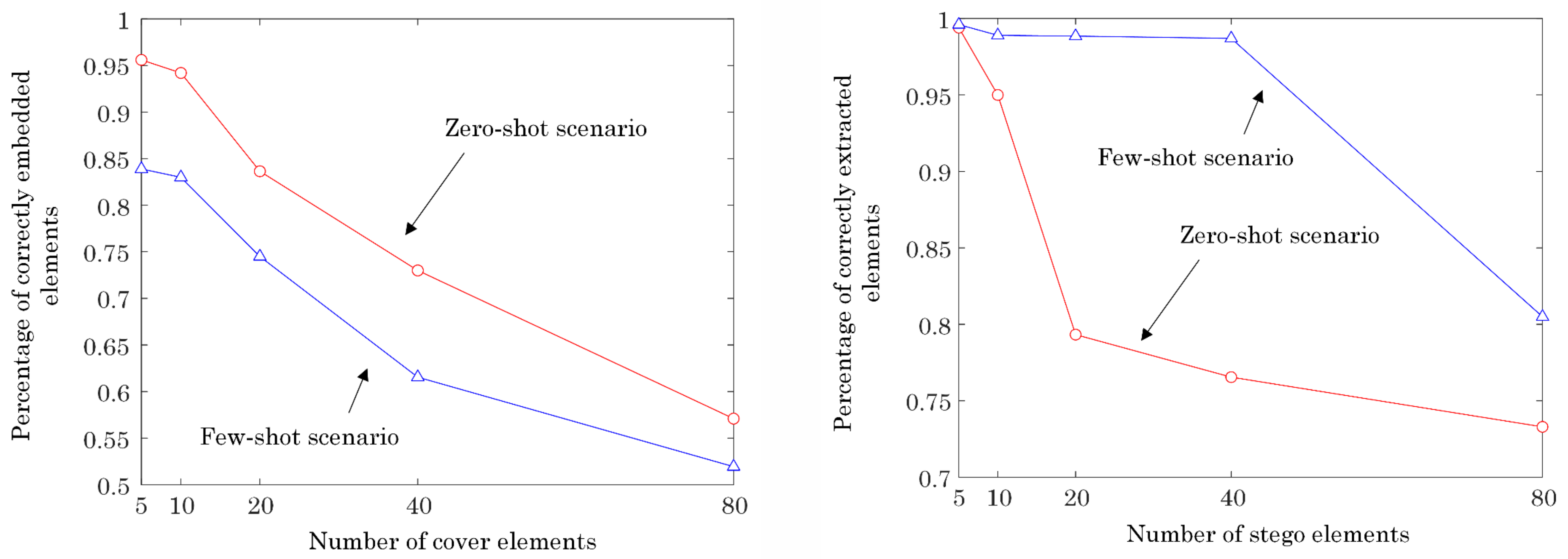
Y-axis: accuracy



2 Prompting Steganography

Prompting Steganography

- How to boost the performance of zero-shot/few-shot steganography?
  - **Optimizing the input prompt**



Outline

1 Introduction

2 Prompting Steganography

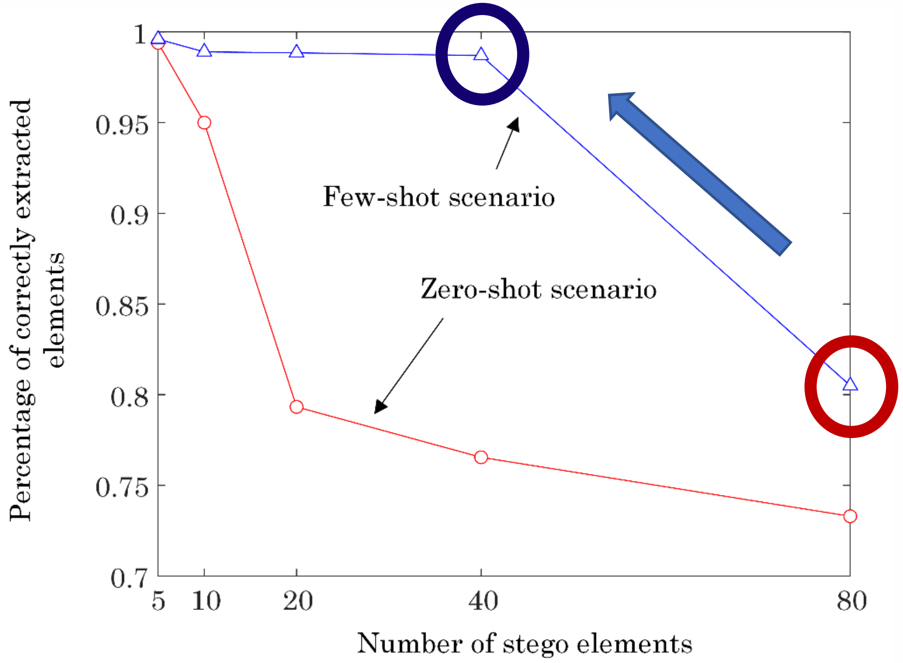
3 Prompt Optimization

4 Conclusion and Discussion

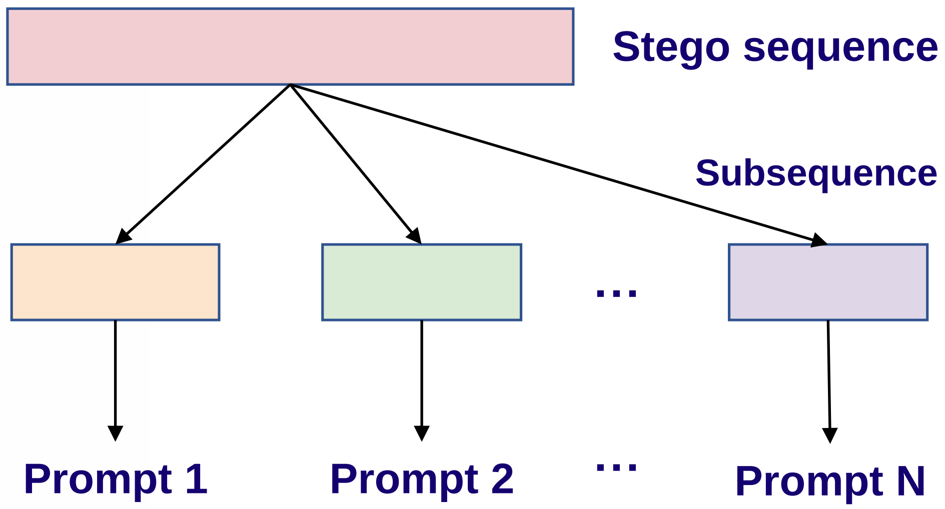
3 Prompt Optimization

❑ Segmentation

- ❑ Divide the original sequence into a certain number of subsequences
- ❑ Each subsequence corresponds to a prompt



(only one prompt)



Stego sequence

Subsequence

…

Prompt 1

Prompt 2

…

Prompt N

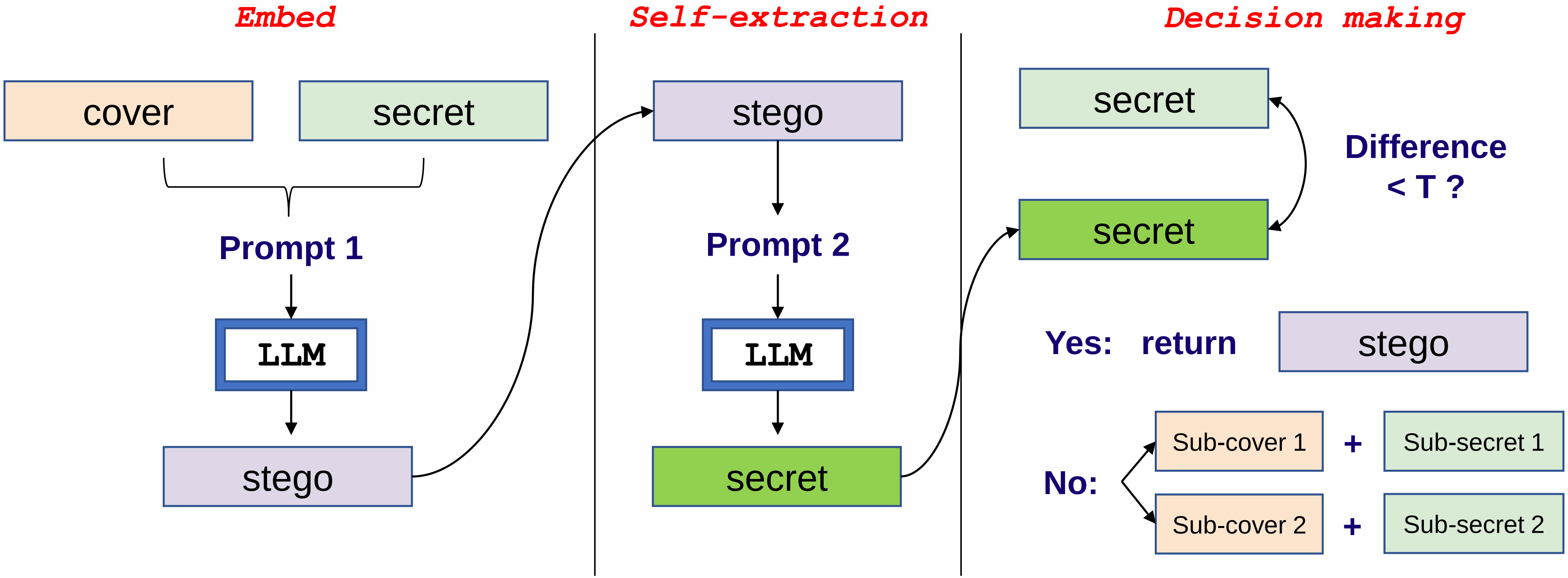
(N prompts)

3 Prompt Optimization

Data embedding	Baseline Acc. + 16% (max)
Data extraction	Baseline Acc. + 19% (max)

☐ Divide and Conquer

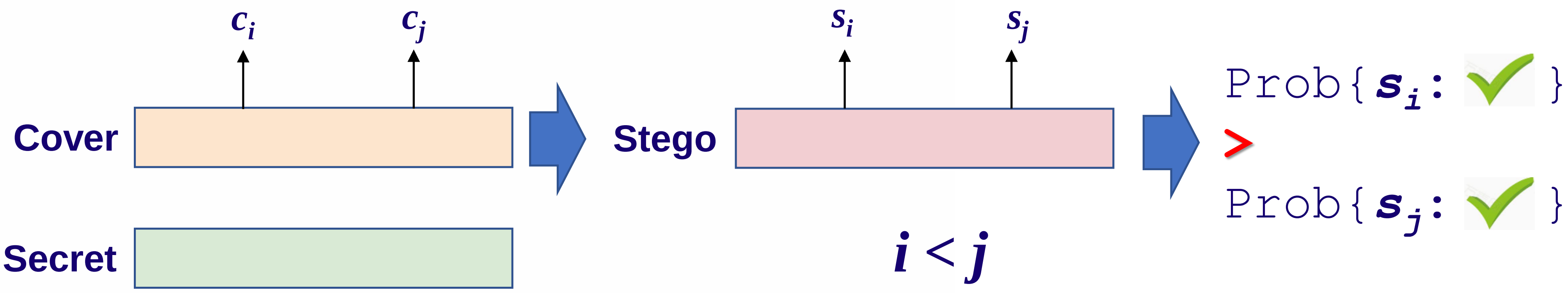
- ☐ Recursively divides the entire sequence into disjoint sub-sequences



3 Prompt Optimization

❑ Position-aware Fusion

- ❑ The steganography performance of the LLM is affected by the positions of the cover (stego) elements in the prompt
- ❑ A cover (or stego) element with a smaller index is more likely to be successfully embedded or extracted



$i < j$

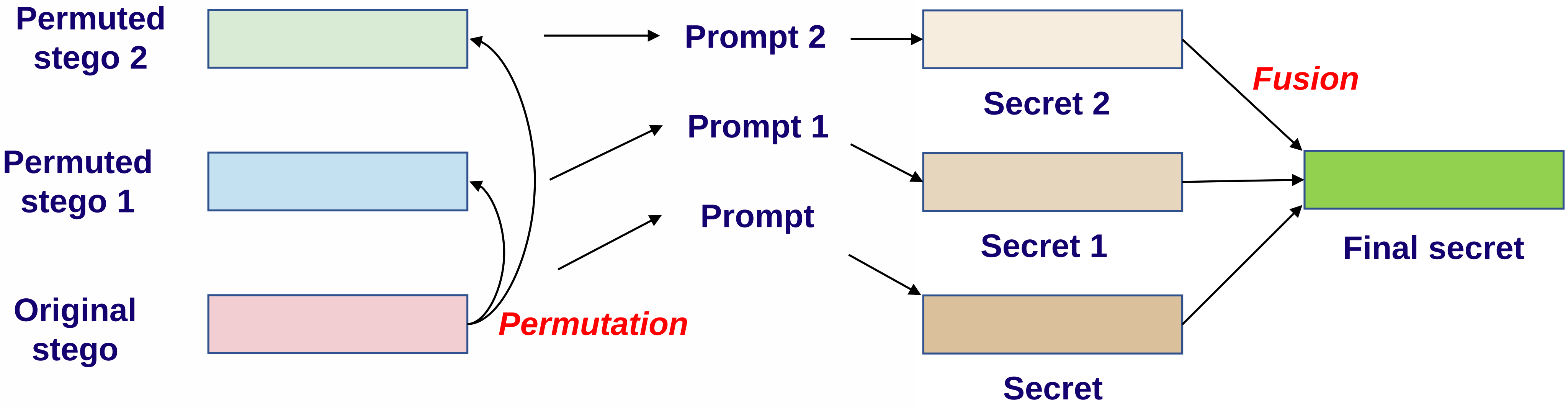
$$\mathrm{Prob}\{s_i: ✓\} > \mathrm{Prob}\{s_j: ✓\}$$

3 Prompt Optimization

❑ Position-aware Fusion

Data embedding	Baseline Acc. + 17% (max)
Data extraction	Baseline Acc. + 25% (max)

- ❑ Generate *multiple* prompts by ***permutation***
- ❑ Merge multiple candidate solutions into the final solution



4 Conclusion and Discussion

❑ Conclusion

- ❑ Prompting steganography is totally different from previous frameworks
- ❑ A pre-trained LLM can embed secret bits into a cover sequence and extract secret bits from a stego sequence, with an error rate
- ❑ This error rate will increase as the number of secret bits becomes larger
- ❑ This error rate can be reduced by optimizing the input prompt

❑ Discussion

- ❑ Future works include *prompt improvement* and *how to guide the LLM to design and implement new steganography algorithms*
- ❑ In future, AI may *replace humans* to develop steganography algorithms

上海大学
SHANGHAI UNIVERSITY

Many Thanks!

Contact email: **h.wu.phd@ieee.org**