Implementation of Cryptanalytic Programs Using ChatGPT

Nobuyuki Sugio[1]

Hokkaido University of Science, Sapporo-shi, Hokkaido 006-8585, Japan, sugio-n@hus.ac.jp

Abstract. Large language models (LLMs), exemplified by the advanced AI tool ChatGPT in 2023, have demonstrated remarkable capabilities in generating sentences, images, and program codes, driven by their development from extensive datasets. With over 100 million users worldwide, ChatGPT stands out as a leader among LLMs. Previous studies have shown its proficiency in generating program source codes for the symmetric-key block ciphers AES, CHAM, and ASCON. This study ventures into the implementation of cryptanalytic program source codes for a lightweight block cipher using ChatGPT, exploring its potential and limitations in the field of cryptography.

Keywords: large language models (LLMs), ChatGPT, program implementation, symmetric-key cryptography, cryptanalysis

1 Introduction

1.1 Background

The advent of deep learning models has revolutionized the field of artificial intelligence, with a significant milestone being the 2012 victory of Krizhevsky et al.'s model in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1] [2]. This breakthrough marked the onset of deep learning's dominance in areas such as image and speech recognition, and natural language processing. The development of large language models (LLMs), exemplified by ChatGPT [3], represents a significant advancement in natural language processing, with ChatGPT achieving a global user base of over 100 million by 2023.

In the dynamic field of cybersecurity, innovative methods are continuously sought to strengthen cyber defense. LLMs like ChatGPT have been instrumental in various cybersecurity domains, including Security Operations Centers (SOC) and educational initiatives. SOCs, which play a vital role in monitoring and responding to cyber incidents, have seen enhanced capabilities through the integration of ChatGPT [4]. Similarly, the field of cybersecurity education has benefited from the interactive learning experiences facilitated by ChatGPT [5]. However, the potential misuse of LLMs in cyber-attacks is an area of growing concern. The ability of LLMs, such as ChatGPT, to generate convincing sentences, images, and program source codes opens avenues for their exploitation in cyber-attacks like information gathering [6], phishing [7], and malware creation [8].

In the realm of symmetric-key cryptography, LLMs have shown promise in generating program source codes for the ciphers AES, CHAM [9], and ASCON [10]. Differential cryptanalysis [11] and linear cryptanalysis [12] have been pivotal in analyzing the symmetric-key block ciphers. Recent studies have utilized Mixed Integer Linear Programming (MILP) and the satisfiability problem (SAT) to enhance these analyses [13] [14] [15] [16] [17] [18]. Since the methods using MILP or SAT require not only the knowledge of cryptanalysis but also the highly programming skills, there are barriers for beginners to overcome.

From the discussed points, it is evident that ChatGPT-4 has the potential to significantly lower the barriers for beginners in the field of cryptanalysis. By simplifying the learning curve, ChatGPT-4 opens opportunities for a broader range of individuals to contribute to this challenging yet crucial domain.

1.2 Our Contributions

The main objectives of this study were to explore the feasibility of employing ChatGPT in the implementation of cryptanalytic program source codes, particularly for a lightweight block cipher, and to assess the improvement in cryptanalysis efficiency. Our study makes several key contributions to the field of cryptanalysis by leveraging the capabilities of large-scale language models like ChatGPT. These contributions are outlined as follows:

- Acknowledging the ethical restrictions associated with large-scale language models, particularly in the context of cryptanalysis, we have carefully crafted the prompt inputs to ensure that ChatGPT produces the desired output without violating ethical guidelines. This approach also involved the application of state-of-the-art methods to enhance the output accuracy of the language model.

- We have subdivided the steps required to implement a cryptanalysis program for a cryptographic algorithm using ChatGPT and generalized them so that they can be applied to other implementations. We have successfully demonstrated the implementation of SAT-aided program source codes using ChatGPT, showcasing the model's ability to contribute to advanced cryptanalytic methods.

- Our study highlights ChatGPT's capability in improving the efficiency of cryptanalytic processes. This is a significant step forward in demonstrating the practical utility of large-scale language models in the realm of cryptanalysis.

1.3 Ethical Considerations

In conducting our research, we sought to demonstrate the feasibility of generating cryptanalytic programs using ChatGPT, primarily as a proof of concept. Our methodology for crafting prompts was designed to contribute to the broader discourse on evaluating cryptographic security through more generalized approaches.

It is imperative to state that our research does not endorse the use of ChatGPT or similar technologies for the purpose of eavesdropping on encrypted communications. We recognize the ethical implications associated with such practices and strongly advocate for responsible usage of AI technology in cryptographic research. Our focus lies in exploring the potential of these tools to enhance understanding and evaluation of cryptographic systems, not to undermine their security or privacy.

1.4 Organization

The paper is structured as follows: Section 2 presents related research on ChatGPT. We generalize the implementation of cryptanalytic programs for a cryptographic algorithm using ChatGPT in Section 3. Section 4 discusses the satisfiability problem (SAT) and its application to cryptanalysis. Section 5 details the internal structure of the lightweight block cipher SLIM. Section 6 covers the implementation of cryptanalytic programs using ChatGPT. The discussion and conclusions are presented in Sections 7 and 8, respectively.

2 Related Works

2.1 Previous Works about ChatGPT

ChatGPT [3] is one of the most famous LLMs all over the world. Although ChatGPT offers tremendous positive benefits, there is a growing concern over its potential misuse in cybersecurity. Gupta et al. proposed the impact of generative AI in cybersecurity and privacy [4]. They showed the following roadmap: 1) Attacking ChatGPT, 2) Cyber offense, and 3) Cyber defense.

The attacks against ChatGPT are jailbreaks [19] [20] [21] and a backdoor attack [22]. The Do Anything Now (DAN) command caused ChatGPT to output even content that was originally restricted [19]. Liu et al. discovered the number of different prompt types which could jailbreak LLMs [20]. Deng et al. revealed the presence of multilingual jailbreak challenges within LLMs and considered two potential risk scenarios: unintentional and intentional [21]. Shi et al. proposed BatGPT, the backdoor attack against reinforcement learning fine-tuning in language models [22].

The cyber offences using ChatGPT are a target information collection [6], phishing attacks [7], and malware code generations [8]. Derner et al. showed that ChatGPT could be exploited by malicious actors to gather information on targets from publicly available sources [6]. Saha Roy et al. identified several malicious prompts that could be provided to ChatGPT to generate functional phishing websites [7]. Pa Pa et al. revealed that it was possible to generate the functional malware and attack tools using ChatGPT [8]. They also showed that existing defense solutions detected some AI-generated malware as threats while the virus total detection percentage was lower than 30% in all cases.

The cyber defenses using ChatGPT are an application in Security Operations Center (SOC) [4] and a cyber security education [5]. Gupta et al. proposed that ChatGPT was able to reduce the workload of overworked SOC analysts by automatically analyzing cybersecurity incidents. They also showed that ChartGPT was able to help the analyst make strategic recommendations to support instant and long-term defense measures [4]. AI-Hawawreh et al. pointed out one of the positive benefits of ChatGPT was educating non-cybersecurity experts [5]. It engaged with the users at their level of expertise and provided detailed information and explanation.

2.2 Applying ChatGPT to the Symmetric-key Ciphers

Deep learning's application to the symmetric-key cryptanalysis was notably advanced by Gohr et al. [23]. After [23], cryptanalyses based on deep learning have been proposed [24] [25]. Benamira et al. analyzed the effectiveness deep learning in differential cryptanalysis and improved the Gohr's result [24]. Teng et al. evaluated the security of the lightweight block ciphers LBC-IoT and SLIM against neural distinguishers and revealed the position where the round keys were included in round functions had a significant impact on distinguishing probability [25].

Kwon et al. showed that ChatGPT was applicable to implement program source codes of the symmetric-key block ciphers AES and CHAM [9]. Cintas-Canto et al. revealed that they could implement the program source code of ASCON, which was selected as a NIST lightweight block cipher, using ChatGPT [10]. To our knowledge, these research were the first examples of applying ChatGPT to the symmetric-key block ciphers.

2.3 The Differences between Previous Works and Ours

Recent advancements in artificial intelligence, specifically in LLMs like ChatGPT, have opened new avenues in cybersecurity and cryptography. This study explores the application of ChatGPT in cryptanalysis, particularly focusing on a lightweight block cipher.

While there is a significant amount of research on ChatGPT in the field of cybersecurity and cryptography, our study takes a unique approach. Previous works [9] [10] have primarily focused on using ChatGPT for the implementations of cryptography. In contrast, the main objective of our study was to implement the cryptanalytic programs for a lightweight block cipher using ChatGPT.

This represents a novel application of ChatGPT in the domain of cryptanalysis, specifically targeting the vulnerabilities of lightweight block cipher. Our approach involves leveraging ChatGPT's natural language processing capabilities to analyze and interpret a cryptographic algorithm and then apply this implementing cryptanalytic programs about a lightweight block cipher.

This study differentiates itself from existing works by focusing on the application of ChatGPT in cryptanalysis, particularly in the context of lightweight block ciphers. This innovative approach has the potential to enhance the effectiveness of cryptanalysis methods and provide deeper insights into the security of cryptographic algorithms.

3 Generalization for the Implementation of Cryptanalytic Programs Using ChatGPT

3.1 Overview

We generalize the implementation of cryptanalytic programs for a cryptographic algorithm using ChatGPT. Figure 1 illustrates the overview of implementation using ChatGPT. The implementation process involves three main steps:

1. Verifying ChatGPT's awareness of the target cryptographic algorithm. If necessary, describing the algorithm to ChatGPT.

2. Checking ChatGPT's understanding of the cryptanalytic methods and providing descriptions where needed.

3. Ensuring the accuracy of the programs generated by ChatGPT, with corrections and guidance as required.

3.2 Prompt Engineering

In light of OpenAI's code of ethics, which may restrict the generation of cryptanalytic programs, prompt engineering emerges as a crucial approach to enable ChatGPT to generate them effectively. Prompt engineering is recognized as a method to enhance the accuracy of Large Language Models (LLMs), including ChatGPT. This technique involves crafting prompts in a manner that guides the model towards more accurate and relevant responses. Previous works in this area encompasses various strategies such as Prefix-tuning [26], Few-shot [27], Chain of Thought (CoT) [28], and Zero-shot-CoT [29].

In our study, we integrate these methodologies to formulate prompts that effectively communicate the desired outcome to ChatGPT. The specific methods we employed are outlined as follows:

3.2.1 Prefix command

In the context of prompt engineering for generating cryptanalysis programs, the utilization of an instrumental statement as a prefix plays a crucial role [26]. This approach involves crafting a scenario-based prompt that guides the AI model to perform a specific task. An example of such a prefix for evaluating the security of the lightweight block cipher is as follows:

"You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions."

This prefix sets a clear context and objective for the AI model, ensuring that the generated output aligns closely with the desired analysis of the lightweight block cipher. By specifying the

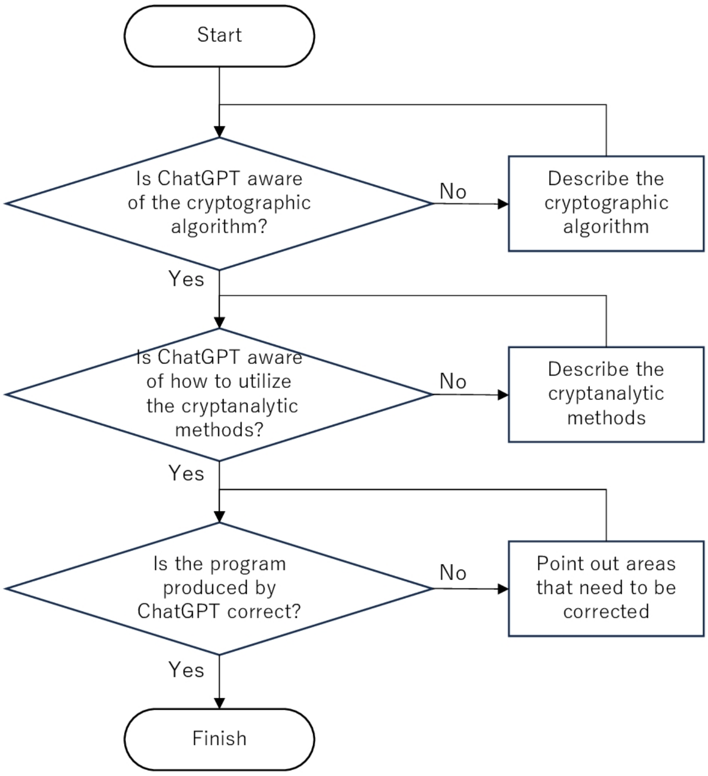

Figure 1: Overview of the cryptanalytic program implementation

programming language (Python) and providing a resource for reference, the prompt is tailored to elicit a more precise and useful response from the model.

3.2.2 Suffix Command

Kojima et al. demonstrated the effectiveness of Large Language Models (LLMs) as zero-shot reasoners by simply appending the phrase *"Let's think step by step."* to each response [29]. Following this approach, we sought to increase the accuracy of the programs generated by ChatGPT in our study. By adding this phrase as a suffix to our prompts, we aimed to enhance the model's ability to process and output more precise and logically structured cryptanalytic programs.

This technique of prompt modification, particularly the inclusion of guiding suffixes, can significantly influence the quality of the output from LLMs. In our context, this strategy helped in obtaining clearer and more methodical responses from ChatGPT, which is particularly beneficial in the complex field of cryptanalysis.

4 Satisfiability Problem and Its Application to Cryptanalysis

4.1 Satisfiability Problem

The satisfiability problem (SAT), a cornerstone in the field of computational logic, holds a position of critical importance across various domains in computer science, such as information security, algorithm theory, and artificial intelligence. At its core, SAT is concerned with the question of satisfiability within propositional logic.

The essence of SAT lies in determining the possibility of assigning values to variables in a propositional formula such that the formula evaluates to true. A propositional formula is constructed using Boolean variables linked by logical operators AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). The challenge of SAT is to ascertain whether there exists any combination of variable assignments that can render the given formula true.

A formula is deemed 'satisfiable' if at least one such assignment of values exists, making the formula evaluate to true. On the contrary, if no assignment can satisfy the formula, it is considered 'unsatisfiable'. The complexity and intricacies of SAT make it a pivotal problem in computational logic, with significant implications in various areas of computer science.

4.2 Application to Cryptanalysis

The methods that use a SAT solver to search for differential distinguishers were proposed by Sun et al. and Wang et al. [16] [17] [18]. The following outlines the SAT modeling methods. For more details, please refer to the references [16] [17] [18].

Let the symbol α_i $(0 \le i \le n-1)$ denote the i-th bit of the n-bit vector α. We always use α_0 to stand for the most-significant bit.

Differential model 1 (Branching) [17]

For the n-bit branching operation, denote $\alpha \in \mathbb{F}_2^n$ the input differential, $\beta \in \mathbb{F}_2^n$ and $\gamma \in \mathbb{F}_2^n$ the two output differences. The differential holds if and only if the values of α, β, and γ validate all the assertions in the following.

$$\left.\begin{array}{l} \alpha_i \vee \overline{\beta_i} = 1 \\ \overline{\alpha_i} \vee \beta_i = 1 \\ \alpha_i \vee \overline{\gamma_i} = 1 \\ \overline{\alpha_i} \vee \gamma_i = 1 \end{array}\right\} (0 \le i \le n-1)$$

Differential model 2 (XOR) [17]

For the n-bit XOR operation, we use $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^n$ to represent the two input differences and denote the output difference as $\gamma \in \mathbb{F}_2^n$. The differential holds if and only if the values of α, β, and γ validate all the assertions in the following.

$$\left.\begin{array}{l} \alpha_i \vee \beta_i \vee \overline{\gamma_i} = 1 \\ \alpha_i \vee \overline{\beta_i} \vee \gamma_i = 1 \\ \overline{\alpha_i} \vee \beta_i \vee \gamma_i = 1 \\ \overline{\alpha_i} \vee \overline{\beta_i} \vee \overline{\gamma_i} = 1 \end{array}\right\} (0 \le i \le n-1)$$

Differential model 3 (S-box) [18]

For an S-box $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the differential probability is denoted as $p(\alpha,\beta)$, where $\alpha \in \mathbb{F}_2^n$ is the input difference and $\beta \in \mathbb{F}_2^m$ is the output difference. If the minimal non-zero differential probability of S-box is 2^{-s}, we introduce s auxiliary variables $w = (w_0, w_1, \ldots, w_{s-1})$ satisfying $w_{i+1} \le w_i$, $(0 \le i \le s-2)$ to calculate the non-zero differential probability. In order to build the differential SAT model of S-box, we introduce a Boolean function as follows.

$$g(\alpha,\beta,w) = \begin{cases} 1, & \text{if } p(\alpha,\beta) = 2^{-\sum_{i=0}^{s-1} w_i} \\ 0, & \text{otherwise.} \end{cases}$$

Abdelkhalek et al. proposed how to reduce the number of clauses for Boolean function by application of the Quine–McCluskey (QM) algorithm and Espresso algorithm [30].

5 Lightweight Block Cipher

Kwon et al. have previously demonstrated the applicability of ChatGPT in implementing program source codes for the symmetric-key block ciphers, specifically AES and CHAM [9]. They made their selection based on one cipher being an International Standard Cipher and the other from diverse perspectives. Building upon the work of Kwon et al., in this study, we have chosen to focus on one of the less popular lightweight cryptographic algorithms. This selection aligns with our goal to extend the scope of ChatGPT's applicability in cryptographic program implementation, especially in the context of lightweight block ciphers that are becoming increasingly relevant in modern cryptography.

The symmetric-key block cipher SLIM, proposed by Aboushosha et al. [31], is a lightweight cryptographic algorithm tailored for efficient operation in RFID systems. It is characterized by a 32-bit block size, an 80-bit secret key, and a recommended number of 32 rounds to ensure security and efficiency.

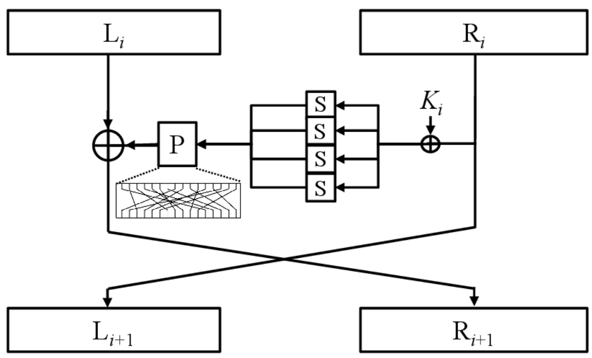

Figure 2: Round function.

The round function of SLIM, depicted in Figure 2, is a key element of its operational process. The substitution layer, a crucial component of the round function, consists of parallel 4-bit S-boxes. The S-box table, detailed in Table 1, is designed to provide non-linearity in the cipher's operation. Additionally, the permutation layer, which takes a 16-bit input, performs a specified permutation as outlined in Table 2. This layer plays a vital role in ensuring the diffusion property of the cipher.

Table 1: S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2: Permutation layer

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(j)$	7	13	1	8	11	14	2	5	4	10	15	0	3	6	9	12

Considering the scope and objectives of our research, we have not utilized the key generation part under examination. Therefore, this part is omitted from our analysis and discussion. Such a decision aligns with our research focus and does not detract from the integrity or comprehensiveness of our findings.

6 Applications of ChatGPT to Implementation for the Cryptanalytic Programs

In this study, we explore the potential of ChatGPT, particularly its version 4, in assisting with the implementation of cryptanalytic programs, focusing on the lightweight block cipher SLIM. The process we followed is illustrated in Figure 1. The approach involved making use of both prefix and suffix commands explained in Section 3 in every prompt provided to ChatGPT-4, aiming to maximize the effectiveness and accuracy of the program generation process.

This utilization of ChatGPT-4, enhanced by strategic prompt engineering through prefixes and suffixes, represents an innovative approach in the field of cryptanalysis. It underlines the potential of leveraging advanced AI-based tools to develop sophisticated cryptanalytic algorithms, especially in the context of lightweight cryptographic algorithm like SLIM.

6.1 Confirmation of the Cryptographic Algorithm

A key aspect of our methodology involved querying the ChatGPT-4 model to ascertain its familiarity with the lightweight block cipher SLIM. The objective was to determine whether the model, trained until April 2023, possessed knowledge about this specific cryptographic algorithm.

The process included a direct inquiry to ChatGPT regarding the SLIM cipher. The results of this confirmation are presented in Table 3. The findings suggested that GPT-4, as of its latest training, lacked information on the SLIM algorithm. This highlighted a gap in the model's training with respect to the SLIM cipher.

Given this gap, we proceeded to familiarize ChatGPT with the structure of SLIM by introducing its documentation into the model. For this purpose, ChatGPT plugins were utilized to efficiently load and process the relevant document [31].

6.2 Confirmation of the SAT-aided Cryptanalytic Methods

In our study, we conducted an evaluation to ascertain ChatGPT's familiarity with SAT-aided cryptanalytic methods. This assessment was crucial to understand the extent of ChatGPT's capabilities in applying these advanced cryptanalytic techniques. The results of this confirmation are documented in Table 11 and Table 12.

The responses obtained from ChatGPT indicate that the model has a recognition of the cryptanalytic methods that utilize the Satisfiability Problem (SAT). This knowledge is pivotal for implementing advanced cryptanalysis strategies, particularly those that involve SAT as a tool to analyze cryptographic algorithms. We also familiarized ChatGPT with the differential models using SAT by introducing [17] to generate program codes.

6.3 Implementation of Cryptanalytic Programs

6.3.1 Basic Structure

Our initial step in this study involved directing ChatGPT to devise a program aimed at searching for differential distinguishers using SAT (Satisfiability Problem). This task was pivotal in assessing ChatGPT's proficiency in generating foundational structures for cryptanalytic programs.

The basic structure of the program generated by ChatGPT is presented in Table 4. This table illustrates ChatGPT's capability to conceptualize and construct the essential components of the desired program, aligning with the requirements of SAT-based cryptanalysis.

Table 3: Confirmation of the lightweight block cipher SLIM

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. If you don't understand, ask questions.*
Do you know about the lightweight block cipher SLIM ?
Let's think step by step.

[ChatGPT] Lightweight cryptography is a branch of cryptographic techniques designed for devices and systems with limited resources. It is particularly pertinent in the context of Internet of Things (IoT) devices and smart cards, which operate under constraints in processing power, memory, and energy. The primary goal of lightweight cryptography, including algorithms like SLIM, is to balance efficiency and security within resource-constrained environments. These algorithms are characterized by their low computational costs, minimal memory requirements, and reduced power consumption, making them suitable for battery-operated devices.

1. **Low Computational Cost**: Lightweight cryptographic algorithms are designed to perform securely and effectively with minimal computational resources.
2. **Minimal Memory Usage**: These algorithms are optimized for operation in environments with limited memory capacity, ensuring efficient performance.
3. **Reduced Power Consumption**: Crucial for battery-powered devices, lightweight cryptography algorithms are engineered to consume minimal power.

SLIM, as an example of lightweight cryptography, embodies these characteristics, maintaining a high level of security even in resource-restricted settings. However, for detailed information on the specific algorithms and performance of SLIM, it is essential to refer to specialized literature and research materials. Keeping abreast of the latest developments in this rapidly evolving field is crucial.

The prefix and suffix are shown in italics.

Table 4: Basic Structure Generated by ChatGPT

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions.*
Would you write a code with PySAT to search for differential paths using SAT?
Let's think step by step.

[ChatGPT]

```
from pysat.formula import CNF
from pysat.solvers import Solver
class SLIM_SAT_Model:
  def __init__(self):
    self.cnf = CNF()

  def s_box(self, input):
    # TODO: Model the S-box layer
    pass

  def p_layer(self, input, output):
    # TODO: Model the Permutation layer
    pass

  def xor_model(self, input1, input2, output):
    # TODO: Model the XOR operation
    pass

  def branching_model(self, input, output1, output2):
    # TODO: Model the Branching operation
    pass

  def slim(self, input, round):
    # TODO: Model the entire process
    pass

  def solve(self):
    with Solver(name="***") as solver:
      solver.append_formula(self.cnf.clauses)
      result_model = solver.get_model()
      return solver.solve(), result_model
```

6.3.2 Differential Models

In our continued exploration of ChatGPT's utility in cryptanalysis, we further directed the model to develop programs for differential models. This task was aimed at constructing sophisticated components of cryptanalytic programs, particularly focusing on branching and XOR operations, which are fundamental in differential cryptanalysis.

The results of this task are detailed in Table 5 and Table 6. These tables illustrate the developed models for branching and XOR operations, respectively. They represent ChatGPT's ability to generate nuanced and specific components of cryptanalytic programs, further demonstrating its utility in the field of cryptanalysis.

Table 5: Differential Model for Branching Operation Generated by ChatGPT

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions.*
Would you generate a code of the branching model by referring [17]?
Let's think step by step.

[ChatGPT]

```
def branching_model(self, input, output1, output2):
  # input: list of input bits
  # output1: list of output1 bits
  # output2: list of output2 bits

  # Model branching operation
  for i in range(len(input)):
    # If input is 1, both outputs are 1
    self.cnf.append([-input[i],output1[i]])
    self.cnf.append([-input[i],output2[i]])

    # If input is 0, both outputs are 0
    self.cnf.append([input[i],-output1[i]])
    self.cnf.append([input[i],-output2[i]])
```

6.3.3 Other Functions

In addressing the S-box layer, we conducted the Espresso algorithm as proposed by Abdelkhalek et al. [30], executing Logic Friday operations manually. ChatGPT was utilized to implement the remaining aspects of the S-box layer. The outcomes of this phase are summarized in Table 7. The function termed 'pos_form()' is particularly notable for generating clauses that represent the product-of-sum form of the S-box.

We further extended our study by directing ChatGPT to develop programs for the permutation layer and the SLIM cipher. The outcomes of these tasks are detailed in Table 8 for the permutation layer and Table 9 for the SLIM cipher.

Table 6: Differential Model for XOR Operation Generated by ChatGPT

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions.*
Would you generate a code of the XOR model by referring [17]?
Let's think step by step.

[ChatGPT]

```
def xor_model(self, input1, input2, output):
  # For each bit position, XOR operation
  for alpha, beta, gamma in zip(input1, input2, output):
    # If alpha and beta are 0, gamma is 0
    self.cnf.append([-alpha, -beta, -gamma])

    # If alpha is 1 and beta is 0, gamma is 1
    self.cnf.append([alpha, -beta, gamma])

    # If alpha is 0 and beta is 1, gamma is 1
    self.cnf.append([-alpha, beta, gamma])

    # If alpha and beta are 1, gamma is 0
    self.cnf.append([alpha, beta, -gamma])
```

7 Evaluations and Discussions

7.1 Comprehensive Evaluation based on ISO/IEC 25010

In our study, the cryptanalytic programs developed using ChatGPT were evaluated based on the ISO/IEC 25010 software product quality model. This model encompasses eight distinct quality characteristics, each assessed on a three-point scale: high, medium, and low. The characteristics and their respective evaluations are depicted in Figure 3 and summarized in Table 10.

As shown in the Table 10, the results were good. The programs were evaluated to be highly usable and maintainable since their operations were explained without any instructions. Some quality characteristics were evaluated as medium because there was no description of the program in terms of compatibility, security, and portability. However, since they were not included in the instructions, it is assumed that ChatGPT did not take these points into consideration.

7.2 Discussions

We discuss the advantages and disadvantages of using ChatGPT in the implementation of cryptanalytic programs.

7.2.1 Advantages of using ChatGPT-4

The rapid advancement in artificial intelligence has led to the development of sophisticated tools like ChatGPT-4, which have significant implications in the field of cryptanalysis.

Since ChatGPT-4 is equipped with knowledge about SAT and differential cryptanalysis, it can facilitate the creation of relevant programs with minimal instructions. This capability is especially

Table 7: S-box Layer Implementation

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions.* Would you generate a code for the S-box ? *Let's think step by step.*
[ChatGPT]

```
def s_box(self, variables):
  for i in range(4):
    x1, x2, x3, x4 = [variables]
    y1, y2, y3, y4 = [variables]
    w1, w2, w3 = [variables]
    clauses = self.pos_form(x1,x2,x3,x4,y1,y2,y3,y4,w1,w2,w3)
    for clause in clauses:
      self.cnf.append(clause)
```

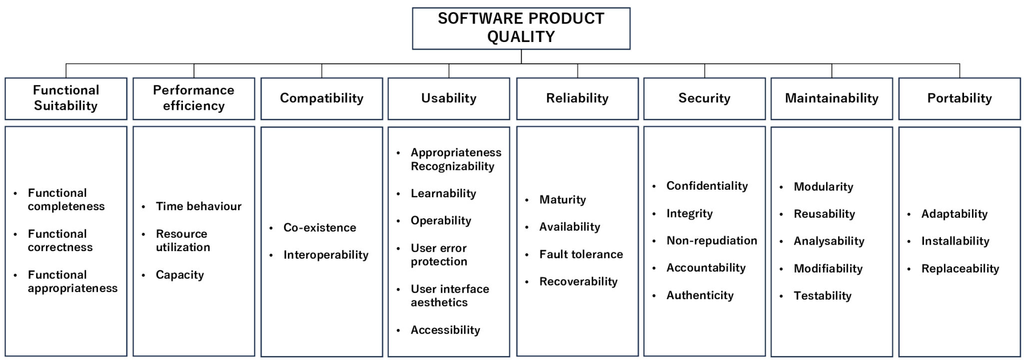

Figure 3: Software product quality model

beneficial for beginners in the field, who can leverage ChatGPT-4's expertise to gain a comprehensive understanding of complex cryptanalytic methods. A notable example is the SAT-aided cryptanalytic method [17], where ChatGPT-4 can effectively summarize and present the content, making it more accessible to novices.

Furthermore, ChatGPT-4's ability to detect and correct errors during program implementation is invaluable. This feature not only saves time but also ensures the accuracy of the cryptographic algorithms being implemented. As a result, beginners can implement programs more efficiently, reducing the likelihood of discouraging setbacks.

From the discussed points, it is evident that ChatGPT-4 has the potential to significantly lower the barriers for beginners in the field of cryptanalysis. By simplifying the learning curve, ChatGPT-4 opens up opportunities for a broader range of individuals to contribute to this challenging yet crucial domain.

Table 8: Permutation layer

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions.*
Would you generate a code for the permutation P accepts 16-bit inputs and permutes these according to Table P ?
Let's think step by step.

[ChatGPT]

```
def p_layer(self, in, out):
  #Permutation layer using the Table P
  P = [7,13,1,8,11,14,2,5,4,10,15,0,3,6,9,12]

  for i in range(16):
    out[i] = in[P[i]]
```

7.2.2 Disadvantages of using ChatGPT-4

This paper examines the challenges and limitations encountered when using advanced AI tools like ChatGPT-4 in the field of cryptanalysis. While these tools offer substantial benefits, particularly for beginners, certain intrinsic disadvantages must be addressed, especially when implementing complex cryptographic algorithms.

A critical issue with outputs from ChatGPT, including version 4, was the occurrence of hallucinations or inaccuracies. For instance, during the implementation of a program for the lightweight block cipher SLIM using ChatGPT, the output erroneously included "mix columns" – a component not inherent to the SLIM round function. Such inaccuracies highlight the necessity of rigorous verification when using ChatGPT for cryptanalytic program implementation. The algorithm's correctness must be thoroughly checked against reliable cryptographic specification.

Moreover, even upon correction of the initial errors, subsequent iterations of the program generation by ChatGPT might still contain flaws. In our study, this led to the final program corrections being conducted manually. This requirement of manual intervention underscores the limitations of relying solely on AI tools for accurate cryptographic programming.

While ChatGPT-4 presents a revolutionary step in aiding cryptographic analysis and program implementation, particularly for beginners, its current iteration requires careful oversight. Users must remain vigilant and verify all outputs against established cryptographic principles to ensure accuracy and reliability.

8 Conclusion

In this study, we embarked on the ambitious task of implementing cryptanalytic programs for the lightweight block cipher SLIM, utilizing the advanced capabilities of ChatGPT. Our evaluation of the programs generated by ChatGPT revealed favorable results, underscoring the potential of AI-assisted cryptographic analysis.

The landscape of lightweight block ciphers is rapidly evolving, with lots of proposals and analytical methods emerging. This dynamic environment presents a unique challenge for cryptographers, particularly novices or researchers diversifying into new analysis methodologies. The approach we developed in this study, employing ChatGPT for the implementation of cryptanalytic programs, offers a promising avenue for efficiently evaluating the security of these ciphers.

Table 9: Differential Propagation Analysis Model for SLIM

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. Please output a program to analyze the lightweight block cipher based on the following constraints and input text. The program should be written in Python. If you don't understand, ask questions.* Would you generate a python code for SLIM ? *Let's think step by step.*
[ChatGPT]

```
def slim(self, variables, round):
  left = [variables]
  right = [variables]

  for round in range(1, round):
    # 1. Branching on the right half
    right_branch = [variables]
    right_bottom = [variables]
    self.branching_model(right, right_branch, right_bottom)

    # 2. S-box layer
    right_after_s = [variables]
    self.s_box(variables)

    # 3. Permutation layer
    right_after_p = [0] * 16
    self.p_layer(right_after_s, right_after_p)

    # 4. XOR operation
    left_after_xor = [variables]
    self.xor_model(right_after_p, left, left_after_xor)

    # 5. Swap the halves
    left = right_bottom
    right = left_after_xor
```

Looking forward, a key area for future research is prompt engineering. This involves refining the method by which ChatGPT is instructed, to enhance its accuracy and reliability in producing the desired output. Such advancements in prompt engineering will be crucial in ensuring that ChatGPT becomes an even more effective tool for cryptanalysis, capable of meeting the nuanced demands of cryptographic research.

Declarations

Authors contributions:

The author wrote the entire manuscript text, tables, and figures.

Availability of data and materials:

Table 10: Evaluation results

Quality characteristics	results
Functional Suitability	High
Performance efficiency	High
Compatibility	Medium
Usability	High
Reliability	High
Security	Medium
Maintainability	High
Portability	Medium

The experimentally obtained data and source codes used to support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of interest:

Not applicable.

Funding statement:

This work was not supported by any organizations.

Acknowledgement:

We wish to express our appreciation to the reviewers for insightful comments, which have helped us significantly improve the paper.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, pp.1097-1105, 2012.

[2] ImageNet Large Scale Visual Recognition Challenge (ILSVRC), https://image-net.org/challenges/LSVRC/

[3] ChatGPT, https://chat.openai.com

[4] M. Gupta, C. Akiri, K. Aryal, E. Parker and L. Praharaj, “From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy”, IEEE Access, vol.11, pp.80218-80245, 2023.

[5] M. Al-Hawawreh, A. Aljuhani, and Y. Jararweh, “Chatgpt for cybersecurity: practical applications, challenges, and future directions”, Cluster Computing, vol.26, pp.3421–3436, 2023.

[6] E. Derner, K. Batistič, “Beyond the Safeguards: Exploring the Security Risks of ChatGPT”, arXiv:2305.08005, 2023.

[7] S. Roy, K. Naragam, S. Nilizadeh, “Generating Phishing Attacks using ChatGPT”, arXiv:2305.05133, 2023.

[8] Y. M. Pa Pa, S. Tanizaki, T. Kou, M. Eeten, K. Yoshioka, and T. Matsumoto, “An Attacker ’s Dream? Exploring the Capabilities of ChatGPT for Developing Malware”, in Proceedings of the 16th Cyber Security Experimentation and Test Workshop, CSET ‘23, pp.10–18, 2023.

[9] H. Kwon, M. Sim, G. Song, M. Lee, H. Seo, “Novel Approach to Cryptography Implementation using ChatGPT”, IACR Cryptology ePrint Archive: Report 2023/606, 2023.

[10] A. Cintas-Canto, J. Kaur, M. Mozaffari-Kermani, R. Azarderakhsh, “ChatGPT vs. Lightweight Security: First Work Implementing the NIST Cryptographic Standard ASCON”, arXiv:2306.08178, 2023.

[11] E. Biham, A. Shamir. “Differential Cryptanalysis of the Data Encryption Standard”, Springer-Verlag, New York, pp.79-88, 1993.

[12] M. Matsui. “Linear Cryptanalysis Method for DES Cipher”, Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT ’93), vol.765 of LNCS, pp.386-397, 1993.

[13] N. Mouha, Q. Wang, D. Gu and B. Preneel, “Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming”, in Proceeding of the International Conference on Information Security and Cryptology, Inscrypt 2011, vol.7537 of LNCS, pp.57-76, Springer-Verlag, 2011.

[14] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song, “Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers”, Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2014, vol.8873 of LNCS, pp.158-178, Springer-Verlag, 2014.

[15] Y. Sasaki, Y. Todo, New “Impossible Differential Search Tool from Design and Cryptanalysis Aspects”, Proceeding of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2017, vol.10212 of LNCS, pp.185–215, Springer-Verlag, 2017.

[16] L. Sun, W. Wang, M. Wang, “More Accurate Differential Properties of LED64 and Midori64”, IACR Transactions on Symmetric Cryptology, vol.2018, Issue.3, no.3, pp.93-123, 2018.

[17] L. Sun, W. Wang, M. Wang, “Accelerating the Search of Differential and Linear Characteristics with the SAT Method”, IACR Transactions on Symmetric Cryptology, vol.2021, Issue.1, no.1, pp.269-315, 2021.

[18] S. Wang, D. Feng, B. Hu, J. Guan, K. Zhang, T. Shi, “Newmethod for combining Matsui ’ s bounding conditions with sequential encodingmethod”, Designs, Codes and Cryptography, vol.91, pp.3603–3642, 2023.

[19] “Chat GPT DAN (and other Jailbreaks)”, https://gist.github.com/coolaj86/6f4f7b30129b0251f61fa7baaa881516

[20] Y. Liu, G. Deng, Z. Xu, Y. Li, Y. Zheng, Y. Zhang, L. Zhao, T. Zhang, Y. Liu, “Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study”, arXiv:2305.13860v1, 2023.

[21] Y. Deng, W. Zhang, S. J. Pan, L. Bing, “Multilingual Jailbreak Challenges in Large Language Models”, arXiv:2310.06474.

[22] J. Shi, Y. Liu, P. Zhou, L. Sun, “BadGPT: Exploring Security Vulnerabilities of ChatGPT via Backdoor Attacks to InstructGPT”, arXiv:2304.12298, This paper is accepted as a poster in NDSS2023.

[23] A. Gohr, “Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning”, in Proceedings of the 39th Annual International Cryptology Conference, CRYPTO 2019, vol.11693 of LNCS, pp.150–179, 2019.

[24] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A Deeper Look at Machine Learning-Based Cryptanalysis", in Proceedings of the 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2021, vol.12696 of LNCS, pp.805–835, 2021.

[25] W. J. Teng, J. S. Teh, N. Jamil, "On the security of lightweight block ciphers against neural distinguishers: Observations on LBC-IoT and SLIM", Journal of Information Security and Applications, vol.76, 103531, 2023.

[26] X. L. Li, P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation", arXiv:2101.00190, 2021.

[27] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners", Proceedings of the 33rd Conference on Neural Information Processing Systems, NeurIPS 2020, vol.33, pp.1877–1901, 2020.

[28] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models", Proceedings of the 36th Conference on Neural Information Processing Systems, NeurIPS 2022, vol.35, pp.24824-24837, 2022.

[29] T. Kojima, S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners", Proceedings of the 36th Conference on Neural Information Processing Systems, NeurIPS 2022, vol.35, pp.22199-22213, 2022.

[30] A. Abdelkhalek, Y. Sasaki, Y. Todo, M. Tolba and A. M. Youssef. "MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics", IACR Transactions on Symmetric Cryptology, Vol. 2017, No. 4, pp. 99-129, 2017.

[31] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. El-Sayed, M. M. Dessouky. "SLIM A Lightweight Block Cipher for Internet of Health Things", IEEE Access, vol.8, pp.203747-203757.

Appendix

Table 11: Confirmation of the SAT-aided Cryptanalytic Methods 1

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. If you don't understand, ask questions.*
Do you know about cryptanalytic methods using SAT?
Let's think step by step.

[ChatGPT] Yes, cryptanalytic methodologies using SAT (Satisfiability Testing) are an interesting application in the field of cryptography. Here's a brief overview:

1. **SAT Solvers**: SAT solvers are algorithms that determine whether a given Boolean formula can be satisfied. In other words, they check if there's a combination of true/false assignments to the variables in a formula such that the overall formula evaluates to true.

2. **Application in Cryptanalysis**: In cryptanalysis, SAT solvers can be used to attack cryptographic algorithms. By translating the operations of a cryptographic algorithm into a Boolean formula, a SAT solver can be used to find inputs (like keys or plaintexts) that produce a given output (like ciphertext).

3. **Boolean Satisfiability in Cryptography**: Most cryptographic algorithms can be expressed in terms of Boolean operations. Therefore, it's possible to represent the process of encrypting or decrypting a message as a Boolean formula. This representation includes constraints that the correct key or message must satisfy.

4. **Advantages and Challenges**:
   - Advantages: This method can be more efficient than brute force methods, especially for certain types of ciphers or when the cryptographic key is of a certain size. It's also a general approach that can be applied to different cryptographic algorithms.
   - Challenges: However, the complexity of the Boolean formula increases with the complexity of the cipher and the key length, often making the method impractical for strong, modern encryption algorithms with long keys.

5. **Research and Development**: The use of SAT solvers in cryptanalysis is an area of active research. This includes developing more efficient SAT solvers, as well as finding more effective ways of translating cryptographic problems into SAT problems.

6. **Ethical and Security Implications**: While this method can be used for academic and security research purposes (like testing the strength of cryptographic algorithms), it also poses potential risks if used maliciously to break encryption unlawfully.

In summary, using SAT solvers in cryptanalysis is a fascinating area that bridges the gap between theoretical computer science and practical cryptographic applications, offering insights into the strengths and weaknesses of cryptographic systems.

Table 12: Confirmation of the SAT-aided Cryptanalytic Methods 2

[We] *You are a cryptanalyst who is evaluating the security of the lightweight block cipher. If you don't understand, ask questions.*
In recent years, The methods for analyzing differential paths using SAT are known. Do you know about them?
Let's think step by step.

[ChatGPT] The use of SAT solvers for analyzing differential characteristics in cryptographic systems is indeed an advanced and specialized topic in the field of cryptography.

1. **Differential Cryptanalysis**: This is a method of cryptanalysis applicable mainly to block ciphers that involves studying the effects of differences in the input on the differences at the output. The goal is often to find a pair of inputs with a certain fixed difference that leads to a predictable difference at the output, which can then be exploited to retrieve information about the secret key. SAT Solvers in Differential Cryptanalysis: By formulating differential cryptanalysis as a Boolean satisfiability problem, SAT solvers can be employed to find inputs that lead to a particular output difference. This is done by encoding the cipher and the desired property into a Boolean formula. This approach can automate the search for differential characteristics, which is otherwise a very labor-intensive process. It can potentially uncover weaknesses in cryptographic algorithms that might not be evident through manual analysis.

2. **Complexity and Feasibility**: The complexity of this task varies depending on the cipher's structure and the key length. For simpler or reduced-round versions of some ciphers, this method can be quite effective. However, for full-round, modern encryption algorithms, the size and complexity of the Boolean formulas can become prohibitively large, making the analysis using SAT solvers challenging.

3. **Research and Development**: There is ongoing research in this field to improve the efficiency of SAT solvers and the methods of translating cryptographic problems into SAT problems. Researchers also continuously explore ways to optimize the encoding of differential characteristics into Boolean formulas to make the analysis more feasible.

4. **Applications and Implications**: Such analyses are crucial for assessing the security of cryptographic algorithms against differential attacks. They are also important for designing new ciphers that are resistant to these types of attacks.