Imperceptible Honey Encryption with Large Language Models

Yanhe Liu, Jiapeng Li, Zhihong Liu
School of Cyber Engineering
Xidian University
Xi'an, China
Email:{liuyanhe,lijiapeng}@stu.xidian.edu.cn

***Abstract*—This paper proposes a honey encryption framework that combines Large Language Models (LLMs) with multiple encoding methodologies, specifically arithmetic coding, self-adjusting arithmetic coding (SAAC), bin coding, and Huffman coding, as a superior alternative to traditional distribution-transforming encoder (DTE). Extensive experimental evaluations indicate that the honey encryption scheme based on the SAAC coding method can produce decoy messages with (1) substantially enhanced linguistic fluency (quantified by perplexity metrics) and (2) significantly improved honey encryption security (measured via Kullback-Leibler divergence), thereby dramatically strengthening the scheme's resilience against cryptanalytic attacks. Furthermore, the novel LLM-driven text transcription mechanism effectively reduces distributional disparities between decoy and authentic messages, providing additional security reinforcement. The investigation also explores practical honey encryption deployment through cloud computing architectures, resolving critical implementation challenges stemming from end-user software/hardware heterogeneity.**

***Index Terms*—Honey Encryption, Large Language Model, Distribution-Transforming Encoder, Self-Adjusting Arithmetic Coding**

I. INTRODUCTION

Privacy preservation constitutes a fundamental requirement for modern online services, particularly in email systems and social networking platforms. Recent years have witnessed escalating threats to user privacy, necessitating robust cryptographic protection mechanisms. In cryptographic systems, sensitive information is encrypted to prevent unauthorized decryption by eavesdroppers. Password-Based Encryption (PBE) serves as a foundational scheme for file encryption and authentication, wherein users simply memorize a passphrase (e.g. password) that the system converts into an encryption key through standardized key derivation functions (KDF) such as PBKDF2, scrypt, or Argon2 [1] . This approach eliminates the complexities associated with conventional key management.

However, existing PBE implementations remain vulnerable to sophisticated attacks. For example, the Dragonfly handshake protocol in WPA3 (a PBE-based PAKE scheme) has been shown to permit offline dictionary attacks due to inherent design flaws that allow password recovery [1] . Furthermore, the pervasive practice of password reuse with weak credentials renders traditional PBE susceptible to brute-force attacks [3]. In the emerging post-quantum cryptographic landscape, conventional algorithms face additional risks of compromise and cryptanalysis [4].

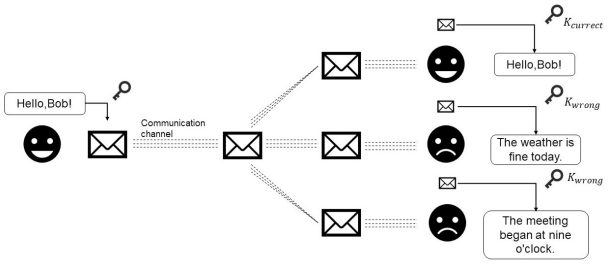

Fig. 1. Honey encryption

Honey encryption represents a cryptographically plausible deniable encryption scheme that prevents adversaries from verifying the authenticity of decrypted content. By design, honey encryption ensures computational indistinguishability between genuine and decoy data, thus increasing the cryptanalytic complexity of brute-force attacks. The core mechanism employs a DTE that converts messages with known probability distributions into uniformly distributed seeds prior to encryption. When decrypted with an incorrect key, the DTE reconstructs semantically plausible decoy messages, causing adversaries to falsely accept invalid keys as authentic. Only legitimate users possessing the correct key can successfully reconstruct the original message's cumulative distribution function and probability density function. Fig.1 illustrates a honey encryption scheme. Decryption with the correct key recovers the original message, whereas an incorrect key yields a semantically coherent decoy message.

However, conventional DTE exhibit three critical limitations: (1) inherent design vulnerabilities that make them detectable through statistical analysis [13]; (2) restrictive applicability to short-form data (e.g., credit card numbers, identification codes); and (3) insufficient semantic coherence in generated decoys. To address these constraints, we propose augmenting DTE capabilities through LLM. LLM has the

capacity to generate extensive and semantically coherent text, extend the length of encrypted messages in honey encryption, and enhance the credibility of decoy messages. Consequently, LLM can replace DTE and compensate for its inherent limitations.

II. BACKGROUND AND RELATED WORKS

In the theoretical study of honey encryption, (Ristenpart et al.) [5] proposed honey encryption using a DTE. By exploiting message distributions, (Li et al.) [19] constructs a standard-model honey encryption scheme with unconditional security and multi-message support, overcoming both Shannon's key-length barrier and prior ROM dependencies.

The applications of honey encryption can be principally categorized as instant messaging and honey password vaults. Within the domain of honey password vaults, Chatterjee et al. [17] improved it with a Natural Language Encoder (NLE) for linguistic deception, and Golla et al. [18] analyzed the role of honey encryption in protecting password managers. These advances demonstrated honey encryption's adaptability in security applications. Cheng et al. [16] enhanced its security using a Probability Model-Transforming Encoder (PMTE), ensuring honey encryption security on applications DTE's working process. Li et al. [8] compromised honey password vaults by exploiting reset mechanisms to distinguish real vaults from decoys, undermining their protection against offline brute-force attacks. Cheng et al. [14] proposed incrementally updateable honey password vaults, extending honey encryption techniques to support dynamic password updates while maintaining security against offline brute-force attacks.

Abiodun et al. [2] proposes an improved honey encryption mechanism to protect user messages by generating plausible-looking fake decoy messages under incorrect decryption attempts, thereby thwarting brute-force attacks.

Regarding the security of LLMs: Zhang et al. [20] established the first unified vulnerability framework for LLMs, distinguishing safety (for example hallucinations), security (for example backdoors), and privacy (for example leakage) threats with corresponding mitigation strategies. Ferrag et al. [21] characterized emerging LLM vulnerabilities (for example adversarial instructions, data poisoning) across the cybersecurity stack, proposing novel mitigations via quantization and human-feedback learning to harden models like Falcon2 and Mixtral-8x7B.

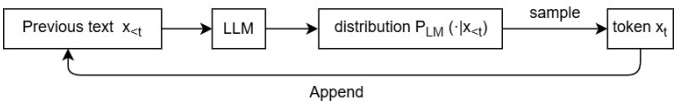

Fig. 2. The operational principle of the LLM.

Large language models (LLM) represent a class of deep neural networks that employ transformer architectures for natural language processing tasks. As illustrated in Fig. 2, the text generation process follows an autoregressive paradigm: Given an input sequence $[x_1, x_2, \cdots, x_{t-1}]$ from vocabulary V, the model computes a conditional probability distribution $P_{LM}(\cdot|x_{<t})$ over the next token x_t. Through iterative sampling from this distribution and concatenation of selected tokens, the model generates syntactically and semantically valid natural language sequences.

The fundamental principle of lossless data encoding lies in the deterministic transformation of information elements into compact bit sequences while guaranteeing perfect data reconstruction. Current implementations employ three principal coding paradigms: (1) Huffman coding [9], which optimizes codeword lengths based on symbol frequency statistics; (2) Arithmetic coding [10], which utilizes recursive interval subdivision for fractional representation; and (3) Bin coding [11], which implements group-based quantization for efficient mapping.

Arithmetic coding [10] constitutes an optimal lossless data compression algorithm that encodes complete input sequences into a single real number within the interval [0,1). The algorithm operates through the following mechanism: (1) initial partition of the interval [0,1) into subintervals proportional to symbol probabilities; (2) iterative interval refinement where each input symbol recursively narrows the current interval to its corresponding subinterval; (3) final representation of the encoded sequence as a binary fraction within the ultimate interval. This method achieves near-optimal compression ratios that approach the Shannon entropy limit, demonstrating particular efficacy for long-sequence data compression scenarios.

Self-adjusting arithmetic coding (SAAC) [12] is an advanced variant of adaptive arithmetic coding that dynamically optimizes symbol probability distributions according to contextual linguistic features. As a cornerstone technique in neurolinguistic steganography, SAAC achieves dual objectives: real-time probability model adaptation based on encoded content characteristics, and near-imperceptible information embedding while preserving textual naturalness. The algorithm's core innovation lies in its context-aware adaptation mechanism, which enables simultaneous optimization of compression efficiency and steganographic covertness, thereby establishing new benchmarks for precision in linguistic steganography.

Bin code [11] operates by first partitioning the vocabulary V into 2^B discrete intervals, where each interval is uniquely represented by a $B-bit$ binary sequence. Subsequently, the ciphertext is segmented into $\lceil L/B \rceil$ contiguous blocks, with LL denoting the total ciphertext length. The t^{th} block is encoded by selecting the most probable word (as determined by the LLM) that falls within the t^{th} interval. This approach ensures efficient mapping between binary representations and linguistic tokens while maintaining compatibility with probabilistic language generation.

Huffman coding is an entropy-based variable length encoding algorithm that constructs an optimal prefix-free binary tree to minimize the expected codeword length. By assigning shorter codewords to high-frequency symbols and longer codewords to low-frequency symbols, this method achieves asymptotically optimal compression rates, approaching the

theoretical lower bound defined by the source entropy [9]. The algorithm's core innovation lies in its greedy tree construction process, which guarantees the uniqueness of decodability while preserving compression efficiency.

III. System Design

As demonstrated in Fig. 3, the system architecture functions as follows: plaintext is used to generate bit sequences through a two-stage procedure. Initially, the LLM derives token probability distributions from the plaintext, and the encoding scheme employs conversion to transform the plaintext into a corresponding bit sequence. This sequence is then encrypted using AES symmetric encryption, yielding the final ciphertext. It is imperative to note that while legitimate users with the correct key K, are capable of successfully decrypting the original message, any decryption attempt employing an invalid key K', will result in the generation of semantically plausible decoy messages through the meticulously designed inverse encoding mechanism.

This paper proposes a novel enhancement to traditional honey encryption by replacing the conventional DTE with an integrated framework combining LLM with multiple encoding schemes (SAAC, bin, huffman, arithmetic). The primary objective is to capitalize on the demonstrated capacity of LLMs to produce high-fluency, semantically coherent text of any length. This advancement significantly improves the quality of decoy messages in honey encryption systems, as LLM-generated alternatives exhibit natural language characteristics that make them computationally indistinguishable from genuine messages to adversaries [6], [7]. The proposed architecture thus establishes a new standard for plausible deniability in cryptographic systems.

The proposed honey encryption architecture adopts a modular design comprising three core functional components: (1) probabilistic encoding, (2) inverse decoding, and (3) cryptographic transformation.

A. Encoding

LLM generates a probability distribution of tokens according to the authentic message, and the encoding scheme maps each token in the authentic message to a probability distribution according to the probability distribution. The encoding scheme then converts the probability distribution of each token into a sequence of bits to complete the encoding.

The encoding process proceeds iteratively as follows: At each step *t*, the encoder computes the conditional probability distribution $Q(y_t|y_{<t})$ using the LLM, and partitions the current interval $[l_t, u_t)$ into sub-intervals, where each sub-interval corresponds to a binary sequence of potential output m_t. The specific sub-interval containing the target value Y_t is then selected to determine the next binary sequence m_t and the updated interval boundaries $[l_{t+1}, u_{t+1})$ for subsequent iterations. This procedure continues until full message generation is achieved, ensuring optimal alignment between the authentic message and the binary sequence.

B. Encryption/Decryption

In the context of the encoded binary sequence, decryption with the correct key reliably reconstitutes the original ciphertext. However, when decrypted with an incorrect key, the decoding process produces a semantically coherent decoy message that exhibits natural language properties indistinguishable from legitimate content, while the derived bit sequence necessarily differs from the original binary sequence.

C. Decoding

The decoding process performs the inverse encoding operation by reconstructing the original message from the bit sequence $m = [m_1, m_2, m_3, m_4 \cdots]$ through iterative interval refinement. The system sequentially processes each m_t, partitioning the current interval $[l_t, u_t)$ into probability-proportional subintervals, and selecting the subinterval corresponding to m_t to obtain updated boundaries $[l_{t+1}, u_{t+1})$. This procedure progressively narrows the interval range until all binary sequence are processed, with the final interval $[l_{t+1}, u_{t+1})$ containing sufficient information to losslessly recover the original message through decoding principles.

IV. Experiment

A. Experiment setting

The experimental configuration employs GPT-2 (345M parameters) as the foundational LLM architecture, utilizing its baseline parameterization without additional modifications. For the encoding schemes, we establish optimized hyperparameters through empirical validation: (a) Bin coding with block size B=3; (b) Huffman coding with tree depth H=7; (c) Arithmetic coding employing K=900 with temperature parameter ζ=1.0; and (d) SAAC with imperceptibility threshold δ=0.01. These parameter selections were rigorously optimized to maximize the fluency and contextual coherence of generated decoy messages, as quantified in subsequent evaluation metrics.

Dataset. COVID-19, which is a subset of research papers related to COVID-19 in the CORD-19 dataset [22], This particular dataset is used in the calculation of D_{KL}.

TABLE I
Parameter Comparison Between Honey Encryption Generated Decoy Messages and Authentic Messages Across Four Encoding Schemes: Smaller PPL and D_{KL} Values Indicate Superior Performance.

Coding methods	**SAAC**	**Huffman**	**bin**	**Arithmetic**
PPL	71.82	131.67	165.69	81.20
D_{KL}	2.6955	3.2076	3.2299	3.1801

B. Results

In our experimental evaluation of the proposed honey encryption scheme, we systematically compare the performance of four coding schemes (SAAC, bin, huffman, arithmetic)

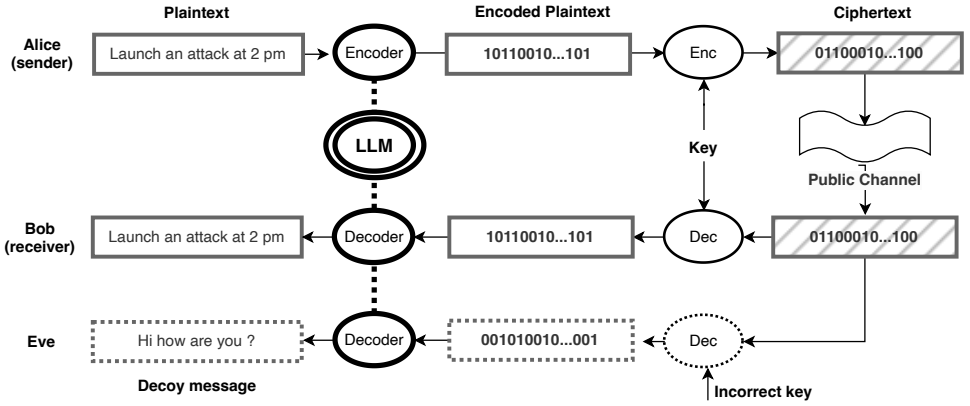

Fig. 3. System architecture

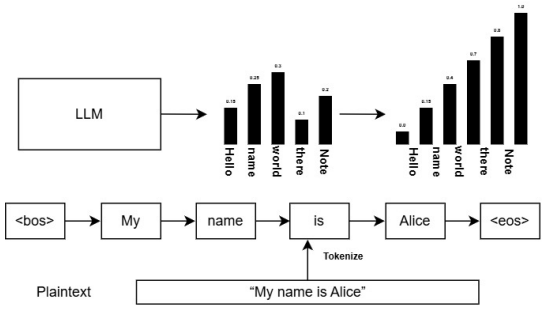

Fig. 4. The process of encoder

using two principal metrics: (1) perplexity (PPL) to assess the linguistic fluency of generated decoy messages, and (2) Kullback-Leibler divergence (D_{KL}) to measure the statistical similarity between decoy and genuine message distributions. These carefully selected metrics provide fundamental insights into the scheme's security characteristics.

Perplexity (PPL) Perplexity quantifies a language model's predictive accuracy by measuring the exponential average of negative log-likelihood per token. Formally defined as PPL, lower PPL values indicate superior model performance, reflecting reduced uncertainty in next-token prediction.

Kullback-Leibler Divergence (D_{KL}): D_{KL} provides an asymmetric measure of dissimilarity between the true distribution P and approximated distribution Q. In steganographic applications, minimized D_{KL} values between decoy and genuine message distributions demonstrate enhanced security through statistical indistinguishability.

The security evaluation of honey encryption requires simultaneous optimization across two interdependent aspects: linguistic quality and statistical distribution. The semantic fluency of decoy messages, quantified through PPL, must achieve parity with genuine messages to prevent adversarial detection of quality discrepancies. Concurrently, D_{KL} between decoy and authentic message (COVID-19) [22] distributions must be minimized to eliminate statistically distinguishable patterns. Security vulnerabilities emerge when either condition is violated, as adversaries can exploit measurable differences in either textual quality or probability distributions to identify honey encryption generated decoys. This dual requirement establishes that effective honey encryption implementations must maintain both high semantic fidelity and distributional indistinguishability to preserve security guarantees against analytical attacks.

The experimental results presented in Table I demonstrate significant performance advantages of the SAAC-based honey encryption scheme over alternative coding methods. Quantitative analysis reveals: (1) The SAAC implementation achieves superior linguistic fluency with PPL of approximately 70, substantially lower than competing schemes, indicating enhanced decoy message naturalness; (2) The D_{KL} for SAAC remains consistently below the critical threshold of 3, outperforming other coding schemes whose D_{KL} values exceed this benchmark. This dual improvement confirms that SAAC-generated decoy messages exhibit both higher semantic quality and better statistical alignment with genuine messages, effectively preventing adversaries from distinguishing between honey encryption processed and natural communications through either linguistic or distributional analysis.

It is important to note that the PPL and D_{KL} metrics for messages processed through honey encryption cannot be directly compared with those of large-volume LLM-generated texts due to their distinct purposes. Honey encryption focuses on securely transmitting short but critical information (typically single sentences or brief paragraphs), while LLM text generation aims to produce extensive, fluent content with maximal informational density. This functional difference explains why honey encryption processed messages naturally exhibit different metric characteristics - the emphasis is on secure transmission of concise information rather than generating lengthy, highly-polished text. The message content in honey encryption remains intentionally compact to serve its security objectives, whereas LLM output prioritizes linguistic excellence at larger scales.

TABLE II
CONTENT COMPARISON BETWEEN AUTHENTIC INFORMATION AND ITS GENERATED DECOY INFORMATION.

Authentic message	**Decoy message**
he says originalists on the court may hew to conservative view, but most of the justices have given clues that they see writing on wall .	To be fair, most New Yorkers who are not on social media aren't ashamed. But isn't public mocking one's new self a pretty deceitful tactic, reliable within U press and state media platforms?
driver, 34, arrested on suspicion of causing death by dangerous driving .	Bodyfat is greatly compromised very very quickly. The basis of volumetric deposition and any long-term goal reduce body weight by 30% to 50% short term, when we experience rapid dietary changes such as increased carbohydrates, body-fat deposition.
hi guys like the title says i'm after bricks but they must be only aussie brands not this overseas shit I'm willing to pay good cash for them contact me on wickr id looking for healing.	BOOK VR HIRLA: REPLACEMENTS who asked for anonymity. It was imperfect. There are several sound effects, but found speakers do sound like they are being jettisoned by Hitchcock.

TABLE III
COMPARATIVE EXPERIMENTS DEMONSTRATE THE DISTINCTIONS AMONG: (I) THE ORIGINAL MESSAGE, (II) THE SEMANTICALLY-PRESERVED TRANSCRIBED TEXT, AND (III) THE DECOY MESSAGE GENERATED FROM THE TRANSCRIBED TEXT REPRESENTATION.

Authentic message	**Transcribed text (LLM)**	**Decoy message**
he says originalists on the court may hew to conservative view, but most of the justices have given clues that they see writing on wall .	He suggests that while the originalists on the court may adhere to a conservative perspective, most of the justices have indicated through their actions that they recognize the inevitable changes ahead.	"I think it's a really good thing to be honest. It's the only way I can get them to take this issue seriously."
driver, 34, arrested on suspicion of causing death by dangerous driving .	A 34-year-old driver has been taken into custody on allegations of causing a fatal accident through dangerous driving.	if you are looking for the "perfect", diet for a healthy body,you should look at the bodybuilding myth.
hi guys like the title says i'm after bricks but they must be only aussie brands not this overseas shit I'm willing to pay good cash for them contact me on wickr id looking for healing.	Hello, I'm searching for bricks, but they need to be Australian-made only—no international brands. I'm ready to pay well for the right products. Reach out to me on Wickr at the ID "looking for healing" if you have what I need.	I'm not sure what the subtitle is. You're right. The movie is a very, very good thriller. But I can't remember what it was about.

TABLE IV
UNDER THE SAAC-ENCODED HONEY ENCRYPTION SCHEME, DECOY MESSAGES ARE GENERATED FROM BOTH (I) MESSAGE BEFORE TEXT TRANSCRIPTION AND (II) MESSAGE AFTER TEXT TRANSCRIPTION VIA LLM PROCESSING, WITH COMPARATIVE EVALUATION BASED ON PPL AND D_{KL} METRICS.

	Message before text transcription	**Message after text transcription**
PPL	71.82	69.13
D_{KL}	2.6955	2.2937

C. The Practical application in real-world network environments

The deployment of honey encryption requires a meticulous examination of both the software and hardware environments. In terms of software configuration, honey encryption operates as a standardized black-box system, in which factors such as hyperparameters[24] and operating systems may have an impact. It should be noted that all users employ uniform LLM parameters and coding schemes, with hyperparameters exerting a comparatively lesser influence on honey encryption. In terms of operating system [23], pycharm, tensorflow, etc. may cause problems with honey encryption results when cross-platform, and also different operating systems may cause honey encryption to run incorrectly, which can have a real impact on honey encryption.

Regarding hardware configurations [25], variations in computational precision across different GPUs and CPUs may lead to inconsistencies in token probability distributions during LLM inference. These hardware-dependent variations can cause the same message to be encoded differently on different

devices, potentially resulting in failed message recovery during decoding. In addition, when users use honey encryption to communicate, different hardware process order and primitive operation will cause problems in honey encryption results, even if single-threaded and serial manner are used to force the control, there will still be problems in honey encryption results due to the floating-point precision and different processing units.

In order to address the challenges mentioned above, a cloud-based implementation is proposed in which honey encryption processing is centralized on servers that are uniformly configured. In this architecture, users submit messages to the cloud server, which handles all honey encryption operations, including encryption and key generation. This ensures consistent encoding/decoding results, regardless of variations in end-user software/hardware.Cloud servers can be regarded as running honey encryption on the same device [25], thereby avoiding many of the aforementioned problems and ensuring the reproducibility of experimental results. The cloud server securely stores ciphertexts and manages key distribution, thereby maintaining the security properties of honey encryption. In this manner, correct keys are required to retrieve genuine messages, while incorrect keys yield plausible decoy messages. This approach guarantees reliable honey encryption operation while supporting large-scale deployment across heterogeneous devices.

V. Semantic-Preserving Text Transcription

Fig. 5 presents the complete text transcription process in our honey encryption system. The experimental results demonstrate that while SAAC-based honey encryption reduces the D_{KL} of the decoy messages, the remaining discrepancy (original $D_{KL} \approx 2.7$) still allows adversaries to identify honey-encrypted decoys, compromising encryption security. To address this vulnerability, we developed a text transcription approach that leverages the inherent flexibility of natural language expression while preserving the core message semantics.

Our solution employs LLMs to rewrite source messages before honey encryption processing, maintaining semantic equivalence while introducing lexical variation. This preprocessing step significantly improves security metrics, as evidenced by the experimental data in Tables III and IV. After text transcription, D_{KL} of the decoy messages decreases from 2.7 to approximately 2.3, indicating a better alignment with the natural language distributions. In particular, this enhancement is achieved without compromising the fluency of the message, as shown by the stable PPL values (±5% variation).

The effectiveness of this method comes from the fundamental properties of natural language. Unmodified messages exhibit higher D_{KL} due to natural linguistic variation, while text transcription constrain this variation without losing expressiveness.

VI. Security Analysis

In this paper, we consider an adversary who understands the honey encryption scheme but lacks the ability to compromise the underlying AES encryption. Through the analysis of decoy information obtained from unsuccessful decryption attempts, the adversary may employ three primary attack strategies:

1) **Statistical Distribution Analysis**
   The adversary compares the decoy and genuine messages using statistical methods. Since decoy messages are generated from the LLM's probability distribution, they approximate genuine messages with high fidelity when contextual information is unknown. This statistical similarity prevents reliable discrimination.
2) **Linguistic Fluency Assessment**
   Perplexity (PPL) comparisons between decoy and genuine messages. Honey encryption's design ensures that encrypted messages maintain concise, high-value content, resulting in minimal PPL differences that thwart manual inspection.
3) **Distributional Divergence Measurement**
   The adversary computes D_{KL} between the distributions of messages. Although there are inherent differences between LLM-generated decoys and original messages, our text transcription solution effectively addresses this vulnerability.

In order to address the vulnerability in distributional analysis, the solution under consideration incorporates LLM-based text transcription prior to the encryption process. This innovative approach maintains the core semantic meaning of original messages while systematically modifying their lexical surface forms. This transformation has been shown to result in a substantial decrease in D_{KL} between text transcription and the original COVID-19 message dataset [22], from 2.7 to approximately 2.3, while ensuring that the message remains fluent, as demonstrated by stable PPL values (±5 percentage point variation). The enhancement of alignment between the statistically optimized message distribution post-transcription and the LLM's inherent output characteristics represents a pivotal improvement.

VII. Conclusions

This paper presents an enhanced honey encryption framework that integrates LLMs with multiple encoding schemes. Our experimental results demonstrate that the self-adjusting arithmetic coding approach (SAAC) significantly enhances both the linguistic fluency (measured by perplexity) and the honey encryption security (quantified by D_{KL}) of decoy messages. Furthermore, the LLM-based text transcription technique effectively minimizes the statistical distribution discrepancies between the decoy and authentic messages. To address practical implementation challenges, we propose a novel cloud-based service architecture that resolves hardware compatibility issues. This research makes significant contributions to the preservation of privacy in the post-quantum cryptography domain. Future work will focus on: (1) optimizing the

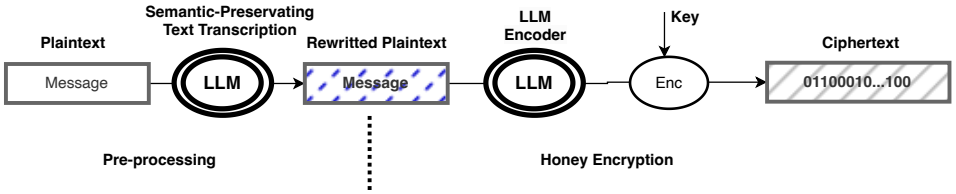

Fig. 5. Semantic-preserving text transcription and honey encryption.

coding-LLM synergy mechanism, (2) developing multimodal honey encryption variants, and (3) employing lightweight LLMs to lower deployment barriers, thereby advancing both the security and practicality of honey encryption systems.

REFERENCES

[1] M. Vanhoef and E. Ronen. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In Proc. IEEE S&P 2020, pages 517-533.

[2] Abiodun, E.O., Jantan, A., Abiodun, O.I. *et al.* Reinforcing the Security of Instant Messaging Systems Using an Enhanced Honey Encryption Scheme: The Case of WhatsApp. *Wireless Pers Commun* 112, 2533–2556 (2020).

[3] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In Proc. ACM CCS 2010, pages 162-175.

[4] Daniel Bernstein and Tanja Lange. Post-quantum cryptography. Nature 549:188-194, 2017.

[5] A. Juels and T. Ristenpart. Honey Encryption: Encryption beyond the Brute-Force Barrier. In IEEE Security & Privacy, 12(4):59-62, 2014.

[6] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Proc. NeurIPS 2019.

[7] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[8] Rao, T., Su, Y., Xu, P., Zheng, Y., Wang, W., Jin, H. (2024). You Reset I Attack! A Master Password Guessing Attack Against Honey Password Vaults. In: Tsudik, G., Conti, M., Liang, K., Smaragdakis, G. (eds) Computer Security – ESORICS 2023. ESORICS 2023. Lecture Notes in Computer Science, vol 14346. Springer, Cham.

[9] Falcon Z. Dai and Zheng Cai. 2019. Towards nearimperceptible steganographic text. In ACL.

[10] Zachary M. Ziegler, Yuntian Deng, and Alexander M. Rush. Neural linguistic steganography. In Proc. EMNLP 2019.

[11] Tina Fang, Martin Jaggi, and Katerina J. Argyraki. Generating steganographic text with lstms. In Proc. ACL 2017.

[12] Zachary M. Ziegler, Yuntian Deng, and Alexander M. Rush. Neural linguistic steganography. In Proc. EMNLP 2019.

[13] H. Cheng, Z. Zheng, W. Li, et al. Probability Model Transforming Encoders Against Encoding Attacks. 2019.

[14] H. Cheng et al., Incrementally Updateable Honey Password Vaults, in *Proc. 30th USENIX Security Symp. (USENIX Security)*, 2021, pp. 857–874.

[15] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751, 2019.

[16] H. Cheng, Z. Zheng, W. Li, et al. Probability Model Transforming Encoders Against Encoding Attacks. 2019.

[17] Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart. Cracking-resistant password vaults using natural language encoders. In Proc. IEEE S&P 2015, pages 481–498.

[18] Maximilian Golla, Benedict Beuscher, and Markus Dürmuth. On the security of cracking-resistant password vaults. In Proc. ACM CCS 2016, pages 1230-1241.

[19] X. Li, Q. Tang, and Z. Zhang, Fooling an Unbounded Adversary with a Short Key, Repeatedly: The Honey Encryption Perspective, in *Proc. 2nd Conf. Inf.-Theoretic Cryptogr. (ITC)*, 2021, pp. 23:1-23:21.

[20] R. Zhang, H.-W. Li, X.-Y. Qian, W.-B. Jiang, and H.-X. Chen, On Large Language Models Safety, Security, and Privacy: A Survey, *J. Electron. Sci. Technol.*, vol. 23, no. 1, p. 100301, 2025.

[21] M. A. Ferrag, F. Alwahedi, A. Battah, B. Cherif, A. Mechri, and N. Tihanyi, Generative AI and Large Language Models for Cyber Security: All Insights You Need, *arXiv*, Art. no. arXiv:2405.12750, May 2024.

[22] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Michael Kinney, Ziyang Liu, William. Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. Cord-19: The covid-19 open research dataset. ArXiv, abs/2004.10706.

[23] Hung Viet Pham, Shangshu Qian, Jiannan Wang, Thibaud Lutellier, Jonathan Rosenthal, Lin Tan, Yaoliang Yu, and Nachiappan Nagappan. Problems and opportunities in training deep learning software systems: An analysis of variance. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, pages 771–783, 2020.

[24] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In NeurIPS, 2018.

[25] O. E. Gundersen, K. L. Coakley, and C. R. Kirkpatrick, Sources of Irreproducibility in Machine Learning: A Review, *arXiv preprint arXiv:2204.07610*, 2022.