Alkaid：通过距离约束编码提升可证明安全隐写对编辑错误的鲁棒性

曹志涵[1], 李高磊[1], 吴军[1],李建华[1],张航[2],和陈明哲[3]l中国上海上海交通大学[2]美国纽约州伊萨卡康奈尔大学[3]美国佛罗里达州科勒尔盖布尔斯迈阿密大学

arXiv:2603.06169v1 [cs.CR] 6 Mar 2026

摘要—**尽管可证明安全隐写通过确保隐写载体与自然样本无法区分来实现强隐蔽性，但这类系统仍容易受到现实世界编辑错误（例如插入、删除和替换）的影响，因为其解码依赖于完美同步，且缺乏纠错能力。为弥补这一不足，我们提出 Alkaid，这是一种通过距离约束编码来抵御编辑错误的可证明安全隐写方案。其关键创新在于将最小距离译码原理直接融入编码过程，对不同消息的码字之间的编辑距离施加严格下界。具体而言，如果两个候选码字违反这一界限，就将它们合并以表示同一条消息，从而保证可靠恢复。在保持可证明安全性的同时，我们从理论上证明 Alkaid 对有界错误具有确定性鲁棒性。为高效实现该方案，我们采用分块处理和批处理。大量实验表明，Alkaid 在各种错误信道中均实现了 99% 至 100% 的解码成功率，提供每个词元 0.2 比特的有效载荷，具有较高的嵌入容量，并保持每秒 6.72 比特的编码速度，在鲁棒性、容量和效率方面显著超越当前最先进水平的方法。**

I. 引言

隐写术是信息隐藏的一项基础技术，其目标是将机密消息嵌入图像、音频信号或文本等表面上无害的载体中，从而实现隐蔽通信 [1]。经典技术，如最低有效位（LSB）嵌入 [2],通过修改像素值的最低有效位来实现这一目标。近年来，基于深度神经网络（DNN）的进展，包括利用生成式对抗网络（GAN）合成载体 [3], [4], [5]以及通过编码器—解码器框架进行自适应嵌入 [6], [7], [8],[9],进一步提升了不可感知性和嵌入容量。然而，尽管方法学持续进步，现有技术仍面临两个内在局限：

(i)* 缺乏可证明的安全保证。** 面对日益复杂的隐写分析对手，大多数现有方法依赖于启发式的不可感知性概念，因此无法提供形式化的复杂性理论安全保证，使隐写载体与自然样本在统计上不可区分这一特性容易受到多项式时间攻击的破坏 [10]。(ii)*对编辑错误的鲁棒性不足。** 在实际部署中，隐写对象经常会遭遇插入、删除或替换操作**（编辑错误）**，这些操作源于裁剪、重新格式化、压缩或传输伪影 。即使是轻微的结构扰动，例如移除图像中的单行或删除一小段文本，也可能破坏嵌入消息的对齐，从而导致无法可靠解码，并严重降低系统可靠性。

为增强安全性，研究者提出了**可证明安全隐写术（PSS）** [12], [13], [14],[15], [16], [17]这一范式。该框架假设存在一个公开已知的概率分布，用于模拟自然载体的统计特征，例如由大语言模型（LLM）或生成式图像模型所诱导的分布。通信双方共享一个密钥，并使用与真正随机源在计算上不可区分的伪随机生成器（PRG），驱动在该分布上的确定性采样过程。具体而言，发送方利用PRG引导的过程，从该分布中确定性地采样出一个载体来编码消息；而接收方使用相同的密钥和PRG，重构采样过程，并通过识别与所接收载体一致的唯一样本来恢复消息。由于PRG与均匀随机性在计算上不可区分，因此生成的隐写载体本身也与从底层分布中抽取的真实样本在计算上不可区分，从而在标准计算意义下实现可证明安全性。

然而，尽管这些方案在形式上提供了强有力的隐蔽性保证，但在现实环境中却表现出明显的脆弱性，因为社交媒体压缩、文本规范化和重采样等常见转换会引入编辑错误，频繁导致解码失败。这种脆弱性源于三个结构性特征：***(i)* 强烈的同步依赖，**因为解码要求精确重现发送方的生成过程；***(ii)* 缺乏错误感知能力，**因此接收方无法检测或定位对载体的修改；以及***(iii)*缺少冗余或纠错结构，**因为现有方案不分配任何冗余，以最大化嵌入容量。

那么，如何在隐写术中实现对编辑错误的鲁棒性，同时保持可证明的安全性？

为解决这一问题，我们提出***Alkaid***，这是一种基于**距离约束编码** 、可证明安全且能够抵御编辑错误的隐写方案。系统的核心在于将最小距离解码原理 [18] 系统地融入隐写编码过程。受此原理指导，我们

发现，解码的鲁棒性从根本上取决于不同消息对应码字之间可度量的距离（例如，汉明距离 [18]或莱文斯坦距离 [19]）。如果序列空间中两个不同消息的码字过于接近，即使是轻微的编辑也很容易造成解码歧义。

为此，我们在编码阶段主动施加严格的序列距离约束。具体而言，在为消息生成候选码字时，算法会动态确保不同消息对应的码字之间的距离不低于预先设定的阈值。如果两个不同消息对应的码字距离低于该阈值，系统便将它们视为相同，并将其编码为同一条消息。该设计保证，只要编辑错误不超过某个界限，接收方就能运用最小距离译码原理，唯一且正确地恢复原始消息，从而实现对编辑错误的容忍。

此外，我们通过形式化证明表明，该系统在保持可证明安全性的同时，还能针对有界编辑错误提供确定性的鲁棒性保证，从而严格统一安全性与鲁棒性。

本文的主要贡献总结如下：

- **理论框架**。 我们提出了一种具有可证明安全性的距离约束编码算法，以解决编辑错误导致的解码失败问题。该算法通过在编码过程中主动引入编辑距离约束，在可证明安全性与抵御编辑错误的鲁棒性之间建立了理论联系。我们通过形式化证明，表明该算法同时实现了信息论安全性和从有界编辑错误中可靠恢复消息的能力。

- **高效实现**。 为使理论框架可操作化，我们提出 Alkaid，一种可证明安全的隐写方案，它通过创新的分块处理与并行化高效实现距离约束编码，从而克服在严格编辑距离约束下编码与解码中的计算瓶颈。

- **全面评估**。大量实验结果表明，Alkaid 优于现有方法，在各种错误信道上实现了 99% 至 100% 的解码成功率，提供 0.2 每个词元的比特数的有效载荷以实现高嵌入容量，并保持 6.72 比特每秒的高效编码速度，从而在鲁棒性、容量和效率方面超越最先进水平（当前最先进水平）方法。

II. 背景与相关工作

A. 预备知识

定义 1（生成模型）。 设 $\mathcal{X}$ 为一个有限离散样本空间，并令 $\mathcal{X}^n$ 表示长度为 n的序列空间。一个生成模型 $\mathcal{G}_\theta$ 是定义在序列空间$\mathcal{X}^n$上、由 $\theta \in \Theta$参数化的概率分布$\mathcal{P}_\theta$。这样的模型可以通过确定性函数Gen_θ： $\mathcal{Z} \to \mathcal{X}^n$实现，其中 $\mathcal{Z}$ 是随机变量的支撑集 ζ，满足当 $\zeta \sim \mathcal{P}_\mathcal{Z}$时，$\mathrm{Gen}_\theta(\zeta)$ 的输出分布恰好为 $\mathcal{P}_\theta$。

一个自回归生成模型 是一类以逐步方式生成序列的生成模型。此类模型定义了序列上的联合分布，并将其分解为

$$\mathcal{P}_\theta(x_1, \ldots, x_n) = \prod_{t=1}^{n} \mathcal{P}_\theta(x_t \mid x_1, \ldots, x_{t-1}). \tag{1}$$

在生成序列 $\mathbf{x} \in \mathcal{X}^n$时，模型根据以下因子分解依次对每个词元进行采样：

$$x_t \sim \mathcal{P}_\theta(x_t \mid x_1, \ldots, x_{t-1}), \quad t = 1, \ldots, n. \tag{2}$$

当前大多数大语言模型（LLM） [20], [21], [22], [23] 采用这种生成范式，通常可以通过指定前缀（或历史）来控制后续内容。

定义2（编辑错误信道）。 编辑错误信道***edit error channel***$\mathcal{E}_e$ 随机地将输入序列 $\mathbf{x} \in \mathcal{X}^n$ 映射为输出序列 $\tilde{\mathbf{x}} \in \mathcal{X}^*$。对于任意输入序列 $\mathbf{x}$，该信道以概率 e在每个位置独立执行一种基本编辑操作（插入、删除或替换），并满足以下期望编辑距离约束：

$$\mathbb{E}[d(\mathbf{x}, \tilde{\mathbf{x}})] \le e \cdot n, \tag{3}$$

其中，编辑距离 （莱文斯坦距离） d（$\mathbf{a}$，$\mathbf{b}$）表示将序列$\mathbf{a}$ 转换为序列$\mathbf{b}$所需的最少基本编辑操作次数，而 $e \in [0, 1]$ 称为该信道的错误率。

定义3（伪随机生成器）。 设 λ为安全参数， G：$\{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ 为一个确定性多项式时间算法，其中 $\ell(\cdot)$ 是一个多项式，且 $\ell(\lambda) > \lambda$。对于任意概率多项式时间（*PPT*）区分器 $\mathcal{A}$，其优势定义为

$$\mathsf{Adv}_{\mathcal{A},G}(\lambda) = \left| \Pr_{\mathsf{sk} \leftarrow \{0,1\}^\lambda} \left[\mathcal{A}(1^\lambda, G(\mathsf{sk})) = 1\right] - \Pr_{\mathcal{S} \leftarrow \{0,1\}^{\ell(\lambda)}} \left[\mathcal{A}(1^\lambda, \mathcal{S}) = 1\right] \right|, \tag{4}$$

其中，概率取自sk 和 $\mathcal{S}$的均匀随机选择，以及 $\mathcal{A}$的内部随机性。此时， G 称为伪随机生成器（***PRG***），如果对于每个概率多项式时间区分器 $\mathcal{A}$，都存在一个可忽略函数 $\mathrm{negl}(\cdot)$，使得$\mathsf{Adv}_{\mathcal{A},G}(\lambda) \le \mathrm{negl}(\lambda)$。

定义 4（生成式隐写方案）。 一个生成式隐写方案 $\Pi = (\mathcal{G}_\theta, \mathcal{M}, \Xi,$ 编码, 解码$)$由以下组件构成：

- 一个生成模型 $\mathcal{G}_\theta$，如定义*1*所述，其输出分布记为 $\mathcal{P}_\theta$；
- 一个消息空间 $\mathcal{M}$，它是一个有限消息集合；
- 一个编码参数空间 Ξ，它是由编码器和解码器共享的一组控制参数；
- 一个编码器Enc： $\mathcal{M} \times \Xi \to \mathcal{X}^n$ 使得，对于任意 $m \in \mathcal{M}$ 和 $\xi \in \Xi$，$\mathsf{Enc}(m, \xi) = \mathsf{Gen}_\theta\big(f(m, \xi)\big)$，其中$f$：$\mathcal{M} \times \Xi \to \mathcal{Z}$是一个确定性函数，将

消息和编码参数映射到输入空间 o f
生成模型，且Gen_θ 是该生成模型的采样函数；

- 满足以下条件的解码器 Dec：$\mathcal{X}^n \times \Xi \to \mathcal{M}$：对于任意 $m \in \mathcal{M}$ 和 $\xi \in \Xi$，$\mathsf{Dec}(\mathsf{Enc}(m, \xi)\ \xi) = m$.

定义 5 (信息论安全性)。设生成式隐写方案 Π 如定义 *4* 所定义，其中消息 m 服从分布 $\mathcal{P}_\mathcal{M}$，且编码参数 ξ 服从分布 $\mathcal{P}_\Xi$。令$\mathcal{P}_\theta$ 表示由生成模型诱导的概率分布 $\mathcal{G}_\theta$。该隐写方案的输出分布记为 $\mathcal{P}_\Pi$，定义如下：

$$\mathcal{P}_\Pi(\mathbf{x}) = \Pr_{\substack{m\sim\mathcal{P}_\mathcal{M} \\ \xi\sim\mathcal{P}_\Xi}} [\mathsf{Enc}(m, \xi) = \mathbf{x}], \quad \forall \mathbf{x} \in \mathcal{X}^n, \tag{5}$$

其中 $D_{\mathrm{KL}}(\mathcal{P}_\Pi \parallel \mathcal{P}_\theta)$表示从 $\mathcal{P}_\Pi$ 到 $\mathcal{P}_\theta$的 ***Kullback–Leibler*** 散度，定义为 *(KL)*

$$D_{\mathrm{KL}}(\mathcal{P}_\Pi \parallel \mathcal{P}_\theta) = \sum_{\mathbf{x}\in\mathcal{X}^n} \mathcal{P}_\Pi(\mathbf{x}) \log \frac{\mathcal{P}_\Pi(\mathbf{x})}{\mathcal{P}_\theta(\mathbf{x})}, \tag{6}$$

它量化了隐写输出分布 $\mathcal{P}_\Pi$ 与载体分布 $\mathcal{P}_\theta$之间的可区分性。若 $D_{\mathrm{KL}}(\mathcal{P}_\Pi \| \mathcal{P}_\theta) \le \epsilon$，则称 Π 为 ϵ-安全。若 $D_{\mathrm{KL}}(\mathcal{P}_\Pi \parallel \mathcal{P}_\theta) = 0$，则称 Π为完美安全。

定义 6（计算安全性）。令 λ为安全参数，并令隐写方案 Π 如定义 *4* 所定义，其所有参数均为 λ 的函数。对于任意概率多项式时间（*PPT*）区分器 $\mathcal{A}$，其优势定义为

$$\mathsf{Adv}_{\mathcal{A},\Pi}(\lambda) = \left| \begin{matrix} \Pr\limits_{\substack{m\sim\mathcal{P}_\mathcal{M} \\ \xi\sim\mathcal{P}_\Xi}} \left[\mathcal{A}(1^\lambda, \mathsf{Enc}(m, \xi)) = 1\right] \\ - \Pr\limits_{\mathbf{x}\sim\mathcal{P}_\theta} \left[\mathcal{A}(1^\lambda, \mathbf{x}) = 1\right] \end{matrix} \right|, \tag{7}$$

并且对于所有概率多项式时间区分器 $\mathcal{A}$，若存在可忽略函数 $\mathrm{negl}(\cdot)$ 使得 $\mathsf{Adv}_{\mathcal{A},\Pi}(\lambda) \le \mathrm{negl}(\lambda)$，则称 Π 为计算安全。

B. 相关工作

1）理论安全隐写：隐写术的形式化安全性定义是其发展的理论基石。传统数字隐写方法主要追求较高的嵌入容量和经验上的不可感知性，却缺乏严格的安全性证明 [24],[25]。Cachin [26] 从信息论角度定义了隐写术的安全性，并以载体分布与隐写分布 [27]之间的 Kullback–Leibler 散度作为衡量指标（定义5对此进行了形式化），从而奠定了隐写分析的理论基础。Hopper 等人 [28]进一步提出了基于计算复杂性理论的可证明安全框架（定义6对此进行了形式化），其中安全性通过概率多项式时间区分博弈来定义：如果对手无法以显著优势区分自然信道分布 O_D 与隐写编码器的输出 Encode_D。这些研究从信息论和计算复杂性两个角度，为此后所有可证明安全的隐写方案确立了严格的安全目标与评估范式。

2）实用安全隐写：在上述理论框架内，研究者致力于构建高效且安全的实用系统。早期的拒绝采样与算术编码方案虽满足安全性定义，但效率低下且难以实际应用 [28], [29], [30]。Meteor [12] 通过对秘密信息进行伪随机化以避免随机性复用问题，在语言生成模型上实现了可证明安全的实用系统，但其嵌入容量受逐步熵浪费的限制。Discop [13] 通过构造多个“分布副本”并利用平衡二叉树进行采样，显著提高了容量，但引入了额外的计算开销。Shimmer [14] 引入了一种熵收集机制，将上一步的残余熵回收到后续编码中，进一步优化了容量与效率之间的平衡。SparSamp [15] 采用稀疏采样策略，在保持安全性的同时追求更高的时间效率。尽管在安全性与容量方面持续取得进展，这些方案仍存在一个根本缺陷：其编码和解码过程必须保持完全同步。对载体序列的任何编辑操作都会破坏这一同步上下文，使解码器无法重构编码路径，从而导致解码失败。这种对编辑错误的严重脆弱性极大地限制了上述方案在真实不可靠信道中的实际应用。

3）鲁棒安全隐写：鲁棒性是确保隐写系统实际可靠性的关键属性。传统的鲁棒隐写研究主要关注抵抗 JPEG压缩、噪声添加和重采样等常规信号处理操作 [31], [32], [33], [34]。其设计通常基于启发式方法或信号处理技术，并未置于可证明安全的形式化框架之中。因此，其安全性难以得到保障。近年来，一些研究开始在追求可证明安全性的同时探索鲁棒性问题。例如，Zamir [35] 探索了将伪随机纠错码 [36] 与隐写术相结合，以增强不可检测性，但未深入分析其对编辑错误的容忍能力。Bai等人提出的面向非对称资源场景的方案 [17] 主要针对抵抗替换错误。STEAD [37]实现了对扩散模型中编辑错误的鲁棒性，但其方法严重依赖特定的模型结构，通用性有限，且嵌入容量较低。因此，在可证明安全隐写的范式下，如何系统地引入对编辑错误的鲁棒性，仍然是一个至关重要但尚未得到充分解决的问题。

III. 距离约束编码

如何构造一种能够抵抗编辑错误的隐写方案？一种可行的方法基于最小距离解码原理：给定接收到的

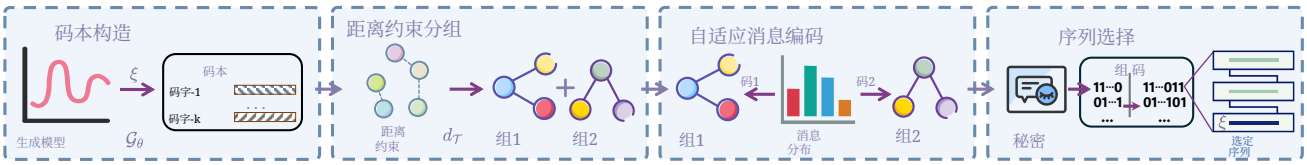


图1. 距离约束编码的过程包括四个主要步骤：***(i)* 码本构造，**其中生成模型 $\mathcal{G}_\theta$ 利用编码参数 ξ 生成 k 条序列作为码字；***(ii)* 距离约束分组，**将编辑距离低于给定阈值 $d_\mathcal{T}$的码字归为一组；***(iii)*自适应消息编码，**在每个组内动态分配码以表示特定消息；***(iv)*序列选择，**其中特定秘密消息与编码参数 ξ 共同确定用作隐写载体的唯一序列。

序列 $\tilde{\mathbf{x}}$ 以及已知的编码参数 ξ，解码器选择与 $\tilde{\mathbf{x}}$在编辑距离意义上最接近的有效码字所对应的消息，其定义如下

$$\mathsf{Dec}(\tilde{\mathbf{x}},\xi)=\arg\min_{m\in\mathcal{M}} d\big(\tilde{\mathbf{x}},\mathsf{Enc}(m,\xi)\big). \tag{8}$$

然而，最小距离解码的有效性依赖于一个重要假设：对应不同消息的码字必须充分分离。如果两个码字距离过近，信道错误很容易使接收序列变得更接近另一个码字，从而导致解码错误。因此，为了确保鲁棒性，我们必须在编码过程中施加明确的距离约束，即对应不同消息的任意两个码字之间的编辑距离不得小于阈值 $d_\mathcal{T}$：

$$d\big(\mathsf{Enc}(m_i,\xi),\mathsf{Enc}(m_j,\xi)\big)>d_\mathcal{T},\quad \forall m_i\neq m_j\in\mathcal{M},\ \xi\in\Xi. \tag{9}$$

为了在保持信息论安全性的同时实现鲁棒性，我们提出了**距离约束编码**。核心思想是在编码过程中施加编辑距离约束，为不同消息分配彼此充分远离的码字，以便解码器能够可靠地纠正信道引入的错误。该过程可概括为以下四个步骤：

A. 主要步骤

码本构造。给定编码参数 $\xi=(\zeta_1,\ldots,\zeta_k,\eta)\in\Xi$，其中 $\zeta_1,\ldots,\zeta_k$用于生成码字，而 η 控制组内选择，我们使用生成模型 $\mathcal{G}_\theta$ 生成初始码本 $\mathcal{C}^\xi=\{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(k)}\}$，其中 $\mathbf{x}^{(i)}=\mathsf{Gen}_\theta(\zeta_i)$。

距离约束分组。设定最小距离阈值 $d_\mathcal{T}>0$。将 $\mathcal{C}^\xi$ 划分为若干组，使得来自不同组的任意两个序列均满足 $d(\mathbf{x},\mathbf{x}')>d_\mathcal{T}$。由此得到的 K 个两两不相交的组为 $\mathcal{O}_1^\xi,\ldots,\mathcal{O}_K^\xi$，并具有如下性质：对于任意 $i\neq j$，$d(\mathbf{x},\mathbf{x}')>d_\mathcal{T}$对所有 $\mathbf{x}\in\mathcal{O}_i^\xi$，$\mathbf{x}'\in\mathcal{O}_j^\xi$均成立。

自适应消息编码。为每个组 $\mathcal{O}_m^\xi$分配唯一的消息 m，得到 $|\mathcal{M}^\xi|=K$。为确保码字的使用概率相等，m 的出现概率应与其所属组的大小成正比，可表示为 $\mathcal{P}_\mathcal{M}^\xi(m)=\frac{|\mathcal{O}_m^\xi|}{k}$。这需要根据分组情况进行自适应消息编码。

序列选择。使用 η 于 ξ，确定性函数h：$\mathcal{M}\times\mathcal{H}\to\mathcal{X}^n$选择码字序列，其中 $\mathcal{H}$是 η的取值空间。该函数满足：对于任意消息 m和编码参数 ξ，当 $\eta\sim\mathcal{P}_\mathcal{H}$时，$h(m,\eta)$保持组 $\mathcal{O}_m^\xi$上的原始概率分布。也就是说，对于任意$\mathbf{x}\in\mathcal{O}_m^\xi$，$\Pr_{\eta\sim\mathcal{P}_\mathcal{H}}[h(m,\eta)=\mathbf{x}]=\frac{t_\mathbf{x}(\xi)}{|\mathcal{O}_m^\xi|}$，其中$t_\mathbf{x}(\xi)$表示$\mathbf{x}$的出现次数。

如图1所示，根据上述步骤，编码器Enc根据消息 m 和参数η，从相应的组中确定并传输码字。解码时，组间强制满足的编辑距离阈值 $d_\mathcal{T}$ 为最小距离解码提供了明确的区分边界，从而在理论上确保了对一定程度信道干扰的鲁棒性。整个方案将结构化码本设计转化为可靠传输的基础。

B. 安全性

定理 1. 设 Π 为一个隐写方案，其中编码器 Enc 使用距离约束编码，且其编码参数 $\xi=(\zeta_1,\ldots,\zeta_k,\eta)$ 由从各自空间中随机采样的相互独立元素组成，并且所有消息空间 $\mathcal{M}^\xi$ 服从分布 $\mathcal{P}_\mathcal{M}^\xi(m)=\frac{|\mathcal{O}_m^\xi|}{k}$，则该隐写方案是完美安全的。也就是说，其输出分布满足 $\mathcal{P}_\Pi=\mathcal{P}_\theta$。

上述定理表明，使用距离约束编码算法生成的隐写文本（隐写文本）不可区分 于模型的原始随机采样输出。完整证明见附录 A-A。

C. 鲁棒性

定理 2. 设 Π 为一个隐写方案，其中编码器 Enc 使用距离约束编码，解码器 Dec 采用最小距离解码。假设编辑错误信道 $\mathcal{E}_e$ 满足定义 2，错误概率为 e。若距离约束满足 $n>d_\mathcal{T}>2en$，则该系统是 (e,δ)-鲁棒 其中

$$\delta\triangleq\exp\left(-\frac{(d_\mathcal{T}-2en)^2}{2n}\right), \tag{10}$$

并且对于任意编码参数 ξ 和任意消息 $m\in\mathcal{M}^\xi$，

$$\Pr[\mathsf{Dec}(\mathcal{E}_e(\mathsf{Enc}(m,\xi)),\xi)\neq m]\leq\delta. \tag{11}$$

该定理保证，当距离约束 $d_\mathcal{T}>2en$ 时，解码失败率的上界较低。完整证明见附录A-B。此外，在实际部署中，样本量 k等参数以及具体的算法实现还可以进一步降低实际失败率，使其低于这一理论阈值。我们将在附录B中分析由此带来的实际鲁棒性。

IV. 主要设计

根据定理1 和2可知，采用距离约束编码并结合最小距离解码的隐写方案，理论上能够同时保证完美安全性和鲁棒性。然而在实践中，这种方法面临三大主要障碍：

***(i)* 生成效率低下。** 码本必须通过独立采样 k 个完整序列来构建，这需要多次调用生成模型，并带来显著的计算开销。

***(ii)* 参数同步方面的挑战。** 编码参数 $\xi = (\zeta_1, \ldots, \zeta_k, \eta)$ 必须由严格独立且分别服从各自分布的元素构成。在分布式环境中，实现大规模参数共享和同步十分困难。

***(iii)*消息空间依赖性。** 消息概率 $\mathcal{P}^{\xi}_{\mathcal{M}}(m) = {}^{|\mathcal{O}^{\xi}_{m}|}$ 同时取决于生成模型 $\mathcal{G}_\theta$和编码参数 ξ。对 ξ 的任何更改，都需要重新编码消息，以保持与相应消息分布的一致性。

为解决上述问题，我们提出了一系列高效实现距离约束编码的方法，显著提升其实用性。下面将详细阐述核心方法。

A. 核心方法

码本构造。为避免显式传输和同步编码参数 ξ，编码器和解码器共享一个密钥sk，并使用密码学安全的伪随机生成器（PRG）即时生成所需的随机性。在实际实现中，我们可以将 PRG 随机生成的序列划分为多个数据块，并将其处理为编码参数。此外，由于大多数最先进的大语言模型都是自回归的，并按顺序生成词元，隐写方案也必须逐词元构造码本。因此，所提出的方案通过顺序生成来构建码本（算法1）。

具体过程如下。在步骤 t，首先使用 PRG 生成一个随机值$r_t \in [0, 1)$。随后通过逆变换抽样（InvCDF）确定输出词元，得到 x_t =InvCDF(r_t)。重复此过程即可生成 k 个长度为 n_l的序列，这些序列共同构成码本。

然而，即使使用密码学安全的 PRG，其输出在统计上也不等同于从真实分布中独立抽取的样本，而仅与这些样本在计算上不可区分。因此，系统不再能实现信息论意义上的完美安全性，后续的安全性分析因而基于计算安全性的概念展开。

距离约束分组。如算法2所示，该分组过程会将所有成对编辑距离不超过阈值 $d_{\min}$ 的元素分配到同一组。实际上，可以使用并查集（DSU）高效实现这一过程。通过迭代评估编辑

算法 1码本构建

输入： 模型 $\mathcal{G}_\theta$，历史 h，样本量 k，块长度 n_l，伪随机生成器 $\mathcal{R}$。

输出： 码本 $\mathcal{C} = \{b_1, \ldots, b_k\}$。

$\mathcal{C} \leftarrow \emptyset$
对 每个 $i \in [1, k]$ **执行**
  $c_i \leftarrow \emptyset$
  上下文 $\leftarrow h$
  对于每一步 $t \in [1, n_l]$ **执行**
    $p(\cdot) \leftarrow$ 下一个词元分布($\mathcal{G}_\theta$，上下文)
    $r_{i,t} \leftarrow \mathcal{R}(i, t)$
    $x_t \leftarrow$ 逆累积分布函数(p，$r_{i,t}$)
    $c_i \leftarrow c_i \,||\, x_t$； ctx $\leftarrow$ ctx $||\, x_t$
  结束 for
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_i\}$
结束 for
返回 $\mathcal{C}$

算法 2距离约束分组

输入：码本 $\mathcal{C} = \{c_i\}_{i=1}^{k}$，阈值 $d_{\mathcal{T}}$。**输出：**组 $\mathcal{O} = \{O_1, \ldots, O_m\}$。1: 并查集。初始化()2: **对于**所有 $1 \le i < j \le k$，**执行**3: **如果** $d(b_i, b_j) \le d_{\mathcal{T}}$，**则**4: 合并$(i, j)$5: **结束如果**6: **结束循环**7: $\mathcal{O} \leftarrow$ 并查集.组件8: **返回** $\mathcal{O}$

计算元素对之间的距离，并将满足约束的元素合并到同一集合中，该算法最终生成一组互不相交的集合，每个集合都对应于距离约束下的一个有效组，从而高效地完成整体划分。

自适应消息编码。在固定的编码参数下，距离约束编码会产生一个具体的消息概率分布 $\mathcal{P}^{\xi}_{\mathcal{M}}(m)$。为实现相应的编码方案，可以在消息空间上设计一种变长码，使原始分布在编码后得以保留。由于 $\mathcal{P}^{\xi}_{\mathcal{M}}(m) = {}^{|\mathcal{O}^{\xi}_{m}|}$ 且 $|\mathcal{O}^{\xi}_{m}|$ 是整数，因此每个消息概率都是$\frac{1}{k}$ 的整数倍。特别是，当 $k = q^{\ell}$时，整个概率空间可划分为 q^{ℓ} 个等概率单元，并且每个单元都可分配一个由大小为 q 的字母表构成的唯一 ℓ 符号码字。每条消息 m 对应 $|\mathcal{O}^{\xi}_{m}|$ 个这样的单元。如果这些单元具有公共前缀，则该前缀可用作消息 m的编码。

例如，令样本量为 $k = 16$，并考虑分布 $[3/4, 1/8, 1/16, 1/16]$，使概率空间被划分为 16 个单元。在二进制编码下，第一个消息的概率为 $12/16$，因此占据了超过一半的单元。因此，无法为它分配任何非空前缀，因为任何这样的前缀所对应的概率都严格小于原消息概率；所以，它必须用空字符串 $\varnothing$表示。

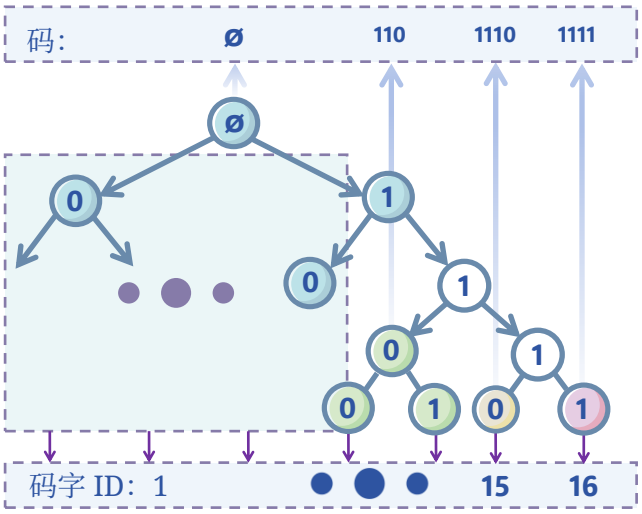


图 2. 自适应消息编码示例：给定组分布 $[3/4, 1/8, 1/16, 1/16]$，我们构造一棵深度为 5 的二叉树。最后一层包含 16 个节点，每个节点代表一个码字。每个组的编码是其码字的公共前缀，最终得到的码为：$[\varnothing, 110, 1110, 1111]$。

接着为第二个消息分配前缀 110，对应接下来的 2 个单元。最后两个消息各占据一个单元，分别编码为1110 和 1111。由此得到的编码为 $[\varnothing, 110, 1110, 1111]$。图 2 生动地展示了这个例子。该构造为同一消息所覆盖的所有单元分配公共前缀，并在解码器处采用最长前缀匹配，从而确保生成的编码分布与原始分布一致。对于诸如 $[3/8, 3/8, 1/4]$这样的分布，同一分配过程会得到编码 $[0, \varnothing, \varnothing]$。在这种情况下，后两个符号共享相同的编码，因此会被明确合并为一个组，并在后续处理中作为整体参与运算。

相应的实现见算法 3。该算法通过动态为不同大小的消息组分配二进制前缀，实现对非均匀消息分布的自适应编码。其核心机制是维护一棵深度为 $\log_2 k$ 的满二叉树，其中每个叶节点对应一个码字，且初始时均未被占用。消息组的大小 $|O|$决定了该组必须占用的叶节点数量 O 。该算法

r

按大小降序处理各组。对于当前组，算法会搜索一个节点 v ，其子树至少包含 $|O|$ 个未占用叶节点，并优先选择深度较大的节点，以获得更长的前缀。随后，它将该子树中的 $|O|$ 个叶节点标记为已占用，并将从根节点到 v 的路径标签（一串 0/1 符号）指定为该组的前缀编码。重复此过程，直到所有组都已分配编码。

序列选择。如算法4所述，输出序列的确定分为两个阶段。首先，每个叶节点代表一个概率单位。给定具有相同码长的秘密消息后，可唯一确定一个叶节点，进而确定其所属的组。其次，通过伪随机生成器生成一个随机值，并将其用作

算法 3自适应编码

输入： 组 $\mathcal{O} = \{O_1, \ldots, O_m\}$，样本量 k。
输出： 消息 $\mathcal{M}$。

```
T ←构建树节点(k)
T.初始化叶节点()
O* ←排序(O)
M ← ∅
对每个组 O ∈ O* 执行
    q ← |O|
    v ← 查找分配节点(T, q)
    p ← 前缀证明节点(v)
    L_O ← 选择叶节点(v, q)
    标记为已用(L_O)
    M ← M ∪ {(O ↦ p)}
结束 for
返回 M
```

算法 4序列选择

输入： 消息 $\mathcal{M}$，秘密比特 s，伪随机生成器 $\mathcal{R}$。
输出： 选定序列 t，已消耗前缀 p。

```
b ← HEAD(s)
O* ← UNITMATCH(M, b)
p ← M[O*]
t ← 逆累积分布函数(O*, R(·))
返回 (t, p)
```

输入到逆变换抽样（逆累积分布函数），从而确定性地产生一条具体序列。根据逆变换抽样原理，只要伪随机数生成器输出是均匀随机的，最终输出便在其对应的组 $\mathcal{O}_m^{\xi}$上遵循指定分布，这确保所得过程构成一个有效的函数 h。

引入伪随机生成器有着关键作用。同一组内的序列彼此相似，若没有同步的随机性，解码器可能无法唯一地恢复发送方所选择的特定序列。通过共享共同种子来同步伪随机生成器，编码器与解码器可以一致地执行相同的组内选择，从而消除解码中的歧义。

B. 概述

分块处理。 构建码本需要预先独立采样 k 条完整序列，这需要反复调用生成模型，并产生相当大的计算成本。当序列长度满足 $n \gg 0$ 和 $k = 2^{\ell}$时，所得计算复杂度为 O（$n{\cdot}2^{\ell}$），其呈指数增长，在实践中往往难以承受。

为解决这一问题，可以采用分块策略。具体而言，将 ℓ比特的消息划分为多个更小的数据块，每个数据块仅包含 b 比特（例如，$b = 1$）。对于每个数据块，仅生成 2^b 个候选序列，然后依次生成并拼接这些数据块，从而得到整体输出序列。如果每个数据块生成长度为 n_l的子序列，则总复杂度可降至$O\big(n_l \cdot 2^b \cdot \frac{\ell}{b}\big)$。选择较小的数值

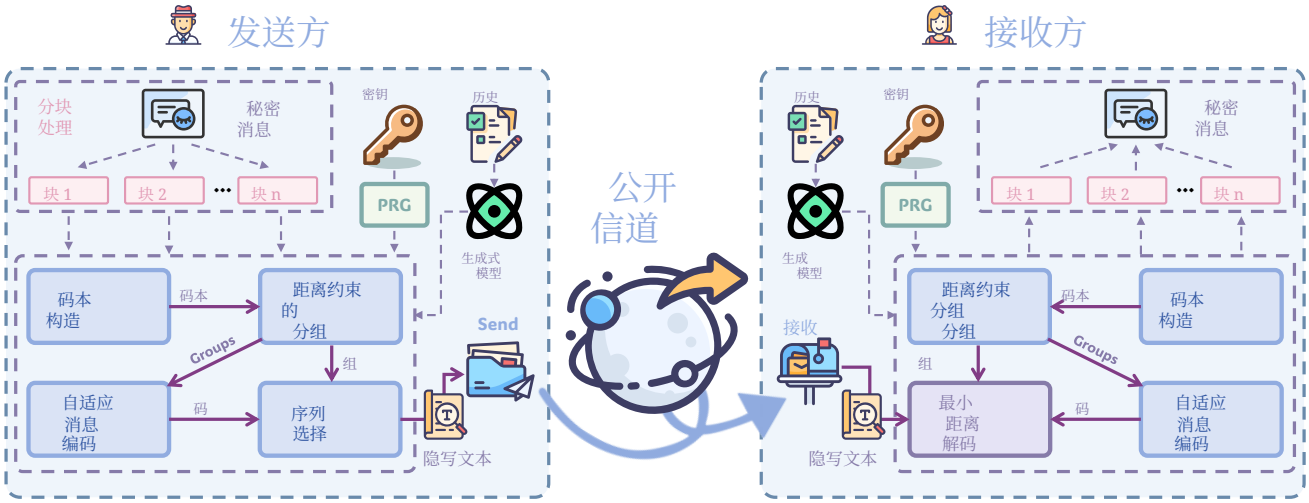


图 3. Alkaid 概览。该方案要求发送方和接收方使用相同的生成模型、历史上下文和密钥。在发送方一侧，秘密消息被划分为多个数据块，每个数据块都经过距离约束编码，直至整条消息嵌入隐写文本中。在接收方一侧，重复相同的距离约束编码过程以重建码本。随后，使用最小距离解码恢复每个消息数据块，最后将其重新组装以获得原始秘密消息。

因此， n_l 和 b 可以显著提高生成效率。

然而，需要注意的是，在基于最小距离解码的纠错设计中，缩短块长度也会相应削弱纠错能力。因此，实际部署时需要根据具体应用场景，在计算成本和鲁棒性之间谨慎权衡。

编码过程。编码过程遵循距离约束编码范式。如图3所示，隐写工作流程按数据块进行。详细过程见算法5。这里，我们假设秘密消息是一个比特串。

首先，将待传输的 ℓ 比特秘密消息划分为多个长度为 b 比特的数据块。对于每个数据块，系统在当前历史上下文的条件下，生成 $k = 2^b$ 个长度为 n_l 的候选序列以构成码本，然后根据编辑距离约束将其划分为多个组。接下来，应用自适应消息编码为每个组分配一个标识符，并根据其内容选出与当前秘密消息数据块对应的组。随后，在选定的组内，利用伪随机性确定性地选出一个特定的输出序列。选定序列会被追加到历史上下文末尾，并作为下一生成轮次的输入。最后，系统计算当前数据块所编码的秘密信息量，并将其从剩余待嵌入消息中移除。本次未能编码的比特将被保留到下一处理周期。该过程持续迭代，直至所有秘密信息都嵌入生成的序列中。

解码过程。解码过程同样按数据块逐块执行，并遵循最小距离解码原则。详细过程见算法6。系统依次处理每个接收到的数据块，并复现相同的码本构造过程

算法5编码

输入：模型 $\mathcal{G}_\theta$**，历史** h**，秘密比特** s**，样本量** k**，距离约束** $d_{\mathcal{T}}$**，块长度** n_l**，种子** $\mathcal{S}$**。输出：编码令牌序列** T**。1：** $T \leftarrow \emptyset$**2：** $\mathcal{R} \leftarrow$ **伪随机生成器.INIT(**$\mathcal{S}$**) 3：while** $len(s) > 0$ **do 4：** $\mathcal{C} \leftarrow$ **码本构建(**$\mathcal{G}_\theta, h, k, n_l, \mathcal{R}$**) 5：** $\mathcal{O} \leftarrow$ **距离约束分组(**$\mathcal{C}, d_{\mathcal{T}}$**) 6：** $\mathcal{M} \leftarrow$ **自适应编码(**$\mathcal{O}$**) 7：(**t**,** p**)** $\leftarrow$ **序列选择(**$\mathcal{O}, \mathcal{M}, s, \mathcal{R}$**) 8：** $T \leftarrow T \,||\, t$**；** $h \leftarrow h \,||\, t$**9：** $s \leftarrow$ **移除前缀(**s, p**) 10：end while 11：return** T

该过程与编码阶段相同，从而获得与当前数据块对应的候选序列集合（码本）。随后，计算接收序列与码本中所有序列之间的距离，并选择包含最近序列的组作为解码结果。

由于使用伪随机生成器（PRG）同步序列选择，重复执行相同过程可确保编码器和解码器之间历史上下文的一致性。为解决插入和删除导致的序列错位，解码器引入大小为w的滑动窗口，扩大比较范围，以减轻此类偏移的影响。同时，在每个数据块内，算法根据解码结果与接收序列之间的匹配情况动态估计偏移量 Δ^* ，从而最大限度地减少偏移误差对后续解码的影响，并减少错误传播。该过程会反复进行，直到整个接收序列完全解码完毕。

算法6解码

输入： 模型 $\mathcal{G}_\theta$、历史 h、接收的令牌序列 $\hat{T}$，样本量 k、距离约束 $d_\mathcal{T}$、块长度 n_l，窗口大小 w ，种子 $\mathcal{S}$。
输出： 重构比特 $\hat{s}$

$\hat{s} \leftarrow \emptyset$, $pos \leftarrow 0$
$\mathcal{R} \leftarrow$ 伪随机生成器.INIT($\mathcal{S}$)
当 $len(\hat{T}) > pos$ **时**
  $\mathcal{C} \leftarrow$ 码本构建($\mathcal{G}_\theta, h, k, n_l, \mathcal{R}$)
  $\mathcal{O} \leftarrow$ 距离约束分组($\mathcal{C}, d_\mathcal{T}$)
  $\mathcal{M} \leftarrow$ 自适应编码($\mathcal{O}$)
  $\tilde{t} \leftarrow \hat{T}[pos: pos + nc + w]$
  $t \leftarrow \arg\min_{c \in \mathcal{C}} d(\tilde{t}, c)$
  $\Delta^\star \leftarrow$ 计算偏移量(t, t)
  $O \leftarrow$ 查找($t, \mathcal{O}$)
  $t^* \leftarrow$ 逆累积分布函数($O^\star, \mathcal{R}(\cdot)$)
  $p \leftarrow \mathcal{M}[O^\star]$
  $\hat{s} \leftarrow \hat{s} \,||\, p$; $h \leftarrow h \,||\, t^*$
  $pos \leftarrow pos + n_l + \Delta^\star$
结束循环
返回 $\hat{s}$

批处理。该 k 码本构造所需的候选序列天然适合并行生成，而非必须严格按顺序逐一执行。利用大语言模型通常支持的批量推理能力 [38]，多个不同的历史上下文 h 可以打包到一个批次中，并在模型推理期间并行处理，从而大幅提高码本生成的效率。

此外，由于生成过程中可能会重复出现相同的模型状态，系统采用缓存机制，直接复用先前计算出的概率分布。这样可以避免对生成模型进行重复访问，并进一步降低整体计算开销。

C. 安全性

定理3. 设 Π是一个隐写方案，其中编码器 Enc采用距离约束编码，解码器 Dec 采用最小距离解码；若编码参数由共享密钥sk通过密码学安全伪随机生成器伪随机生成器导出，则在伪随机生成器相对于安全参数 λ而言与均匀随机源计算不可区分的条件下，系统 Π 是计算安全 的。

上述定理证明，当采用密码学安全的伪随机生成器（PRG）时，Alkaid 生成的隐写文本在计算上不可区分于模型原始随机采样的输出。完整证明见附录A-C。实际的隐写测试结果见附录C。

V. 评估

A. 实验设置

生成模型设置。我们关注当今常用的主流大语言模型（LLM）。我们的实验主要在 Qwen2.5-7B [39], LLaMA-3-8B [40],GLM-4.5-9B [41],和 Mistral-7B [42],上进行，其中 Qwen2.5-7B 作为基线评估的默认模型。在生成过程中，我们在每一步都采用核采样（Top-p） [43]，默认值为 $p = 0.95$。具体而言，由于 STEAD [37] 仅适用于扩散大模型，因此我们为其配置的模型是 Dream-7B [44]。

硬件。 所有实验均在一台配备 25 个 vCPU、Intel(R) Xeon(R) Platinum 8470Q 和 RTX 5090（32GB）的服务器上进行。

指标和评估方法。 我们从四个关键维度评估所提出的方法：

(i) 鲁棒性：通过解码成功率衡量。只有解码后的消息与原始消息完全一致时，才视为成功。我们还报告 $d_{\min}$，即所选码字与码本中备选组内所有其他码字之间的最小编辑距离。$d_{\min}$ 越大，表示容错能力越强。

(ii) 效率：通过编码和解码吞吐量衡量，以比特每秒（比特/秒）和每秒令牌数（令牌/秒）报告。数值越高，表示效率越高。

(iii) 容量：通过有效载荷（比特/词元）和熵利用率衡量。数值越高，表示隐写容量越大。

(iv) 可用性：主要通过困惑度（PPL）衡量。PPL 越低，生成的隐写文本语言质量越高。此外，我们还测量最大 GPU 显存消耗，并通过在多种语言和模型上部署系统，展示其在各种真实场景中的实际适用性。此外，我们还在附录 D 中探讨了使用不同生成模型的情况。

B. 主要结果

Alkaid 能够抵御编辑错误。默认情况下，我们假设系统在错误传输信道中遇到的错误率为 0.15，并采用大小为 $n_l = 20$ 的数据块。因此，阈值为 $d_\mathcal{T} = 0.15 * 20 = 6$。此外，我们选择样本量 $k = 32$ ，以平衡容量和效率。针对每种错误率，我们在编辑错误信道上进行了 10,000 次实验。

如图所示4，在随机编辑错误率为 0.05 至 0.4 的范围内，Alkaid 的表现始终优于所有其他方法。错误率不超过 0.2 时，其成功率保持在 99.6% 以上；错误率为 0.3 时，成功率仅略微降至 98.0%；即使在测试的最高错误率 0.4 下，成功率仍高达 92.6%。相比之下，最先进方法（SOTA）FDPSS [16], SparSamp [15],ARS [17] 的成功率迅速下降，几乎归零。尽管 STEAD [37] 能够抵抗编辑错误，但其在错误率为 0.05 时的成功率仅为 26.5%，且准确率会随着错误率上升而显著下降。这些结果表明，与现有方法相比，Alkaid 对随机编辑错误具有很强的鲁棒性。

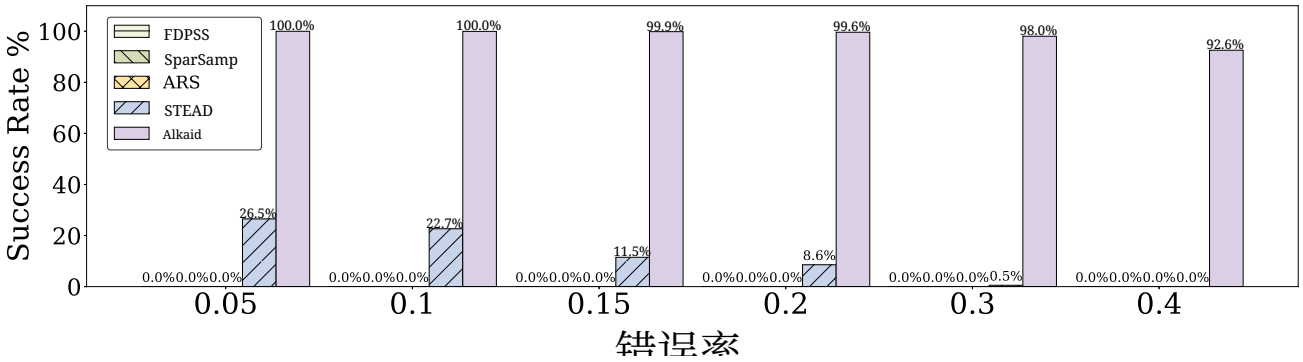


图4. 编辑错误信道 $\mathcal{E}_e$上不同错误率下的解码成功率对比。

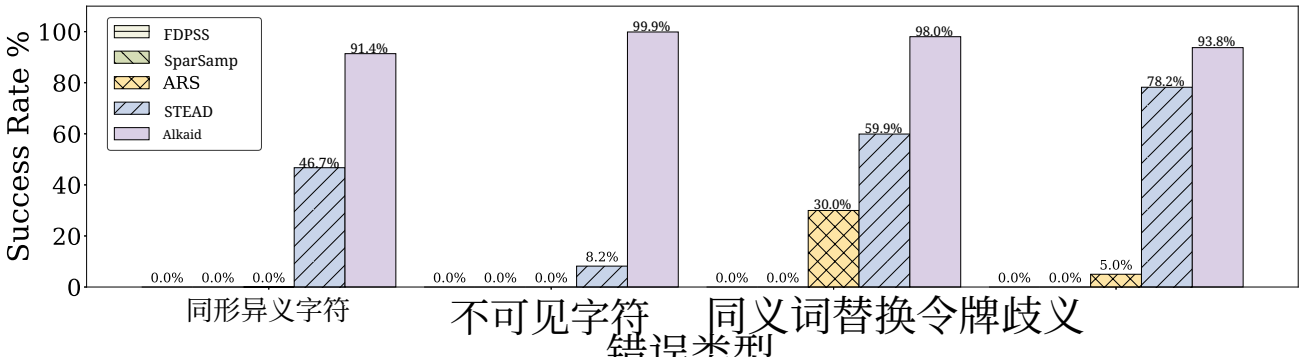


图5. Dec不同令牌级错误下的解码成功率对比 。

Alkaid 能够抵御令牌级错误。与以往研究 [45], [46], 类似，我们进一步针对现有文献中常见的四种令牌级错误开展了实验，其核心操作如下：***(i)*** 同形异义字符，即用视觉上相似的字形替换字符；***(ii)*** 不可见字符，即在文本中插入难以察觉的特殊字符；***(iii)***同义词替换，即用原词的同义词替换原词；***(iv)*** 令牌歧义，即误导分词器将文本解码为错误的词元。在所有实验中，每种错误造成的扰动均不超过目标文本中词元总数的 10%。

如图5所示，Alkaid 在所有四种令牌级错误下都表现出始终如一的高鲁棒性：同形异义字符错误下的成功率为 91.4%，不可见字符错误下为 99.9%，同义词替换错误下为 98.0%，令牌歧义错误下为 93.8%。相比之下，FDPSS 和 SparSamp 表现不佳，完全失效（成功率为 0 %）。ARS 在几乎所有条件下也都彻底崩溃，成功率为 0% 或接近 0%；由于其对词汇替换具有内在的抵御能力，仅在同义词替换错误下取得了微弱的性能。STEAD 表现出中等程度的抵御能力，在同义词替换错误下的成功率为 59.9%，在令牌歧义错误下为 78.2%，但在所有场景中仍远远落后于 Alkaid。这些结果证实，与所有最先进的方法相比，Alkaid 对多种令牌级扰动具有显著更强的鲁棒性。

Alkaid在效率和隐写容量方面优于其他鲁棒隐写。结果如表 I所示。尽管STEAD因其采用Dream-7B扩散模型架构而实现了领先的词元编码速度，达到71.533每秒令牌数，但其实际隐写编码效率仅为2.6385比特每秒，载荷容量低至0.0369每个词元的比特数，熵利用率不足1%，因而整体隐写能力有限。ARS设计了高效的解码流程，解码速度达到106.087比特每秒，但其编码效率仅为0.257比特每秒，载荷容量为0.0136每个词元的比特数，同样存在容量利用率低的问题。相比之下，Alkaid的编码效率达到6.7236比特每秒，远超其他方法。其载荷容量达到0.2045每个词元的比特数，熵利用率为18.47%，表明其能够在单位时间内隐藏更多信息，并显著提高文本熵的利用效率。在可用性方面，所有方法均保持合理的文本困惑度，确保模型功能不受影响。综上所述，Alkaid在保持良好可用性的同时，在编码效率和隐写容量方面均优于其他方法，因此可以得出结论：其在鲁棒隐写的效率与容量上实现了双重领先。

Alkaid 在各种配置下都展现出很高的可用性。如表 II所示，Alkaid 在各种采样配置下展现出稳定且适应性强的性能，这一点得到了四种典型大语言模型的全面性能数据支持：Qwen、Mistral、Llama 和 GLM。默认情况下，我们将配置参数设置为

表I 可证明安全的鲁棒隐写术的效率与容量比较。

方法	效率			容量		可用性	
	编码↑（令牌/秒）	编码↑（比特/秒）	解码↑（比特/秒）	有效载荷↑（比特/词元）	熵↑ 利用率%	熵	PPL↓
STEAD	**71.5330**	2.6385	0.4889	0.0369	0.72%	5.0632	40.1805
ARS	18.6490	0.2570	**106.087**	0.0136	0.71%	1.6500	1.1260
Alkaid	33.0203	**6.7236**	6.7011	**0.2045**	**18.47%**	1.1493	2.4458

表 II 不同模型和采样配置下 Alkaid 的全面评估。

模型	Top-p	鲁棒性		效率			容量		可用性		
		$d_{\min}$↑	成功。↑ 速率%	编码↑（词元/秒）	编码↑（字节/秒）	解码↑（字节/秒）	有效载荷↑ (B./T.)	熵↑ 利用率%	GPU↓ (GB)	熵	PPL↓
QWEN	0.8	10.547	99.9%	35.212	6.086	6.065	0.174	21.0%	16.468	0.881	1.973
	0.9	11.312	100%	34.000	6.474	6.452	0.192	18.0%	16.468	1.145	2.495
	0.95	11.815	100%	33.280	6.591	6.570	0.199	16.0%	16.468	1.357	3.101
	1.0	12.428	100%	18.675	3.857	3.849	0.207	13.3%	16.468	1.703	4.898
MISTRAL	0.8	9.734	99.9%	35.136	4.122	4.108	0.119	22.2%	25.101	0.571	1.632
	0.9	10.151	100%	33.660	4.865	4.849	0.147	20.9%	25.101	0.769	2.048
	0.95	10.442	100%	33.021	5.057	5.041	0.155	19.3%	25.101	0.888	2.538
	1.0	10.943	100%	29.432	4.995	4.981	0.171	15.9%	25.101	1.226	5.539
LLAMA	0.8	9.109	99.9%	33.182	3.464	3.452	0.105	21.9%	25.101	0.474	1.421
	0.9	9.631	99.9%	32.004	4.189	4.176	0.132	21.1%	25.101	0.626	1.594
	0.95	9.984	100%	31.183	4.593	4.579	0.149	20.0%	25.101	0.758	1.812
	1.0	10.221	100%	20.307	3.348	3.341	0.167	17.8%	25.101	0.969	2.456
GLM	0.8	12.476	100%	23.417	4.799	4.788	0.204	12.5%	25.101	1.742	4.666
	0.9	13.587	100%	22.972	4.933	4.922	0.214	9.7%	25.101	2.408	9.335
	0.95	14.368	100%	22.332	4.932	4.921	0.221	8.0%	25.101	3.009	20.556
	1.0	15.037	100%	14.359	3.186	3.181	0.222	6.1%	25.101	3.922	66.441

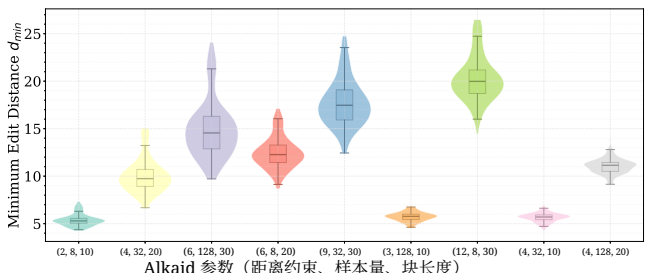
$(d_{\mathcal{T}},\ k,\ n_l) = (6,\ 32,\ 20)$。在鲁棒性测试中，我们将编辑错误信道的错误率配置为 0.15。在所有模型以及从 0.8 到 1.0 的 Top-p 设置下，该系统在每秒比特数方面都保持了较高的编码速度；例如，在早期比较中，Alkaid 的峰值达到每秒 6.723 比特，同时在 GLM 的 Top-p 为 1.0 等配置下，有效载荷容量最高可达每个词元 0.222 比特。更重要的是，在几乎所有场景中，消息恢复成功率都达到 100%；许多设置下的最小距离通常高于 10，这表明系统对编辑错误具有很强的鲁棒性。

不同配置呈现出明显的规律：随着 Top-p 增大，载荷容量往往略有上升，而熵利用率通常会下降。例如，在 Qwen 模型中，Top-p 从 0.8 增至 1.0 时，有效载荷从每个词元 0.174 比特增至 0.207 比特，但熵利用率从 21.0% 降至 13.3%。尽管容量有所降低，这表明鲁棒性有所增强，这一点也反映在 $d_{\min}$ 从 10.547 升至 12.428 上。每秒令牌数表示的编码速度通常保持稳定，直到 Top-p 接近 1.0 时才会明显下降，例如 Qwen 的编码速度从每秒 33 个以上词元降至 18.675。以困惑度衡量的文本自然度在大多数设置下都处于实际可接受的范围内，低于 5.0 对于各模型，当 Top-p 不超过 0.95 时，困惑度均低于该值；但在 Top-p 为 1.0 时，困惑度可能显著上升，某些模型（如 Glm）尤为明显。总而言之，Alkaid 展现出很高的实际可用性和可靠的适应能力，能够在不同模型和采样参数下有效平衡编码速度、隐写容量与文本质量。它在成功率和鲁棒性方面表现稳定，同时在不同条件下保持效率与容量趋势平稳，这证实了它适用于广泛的应用场景。

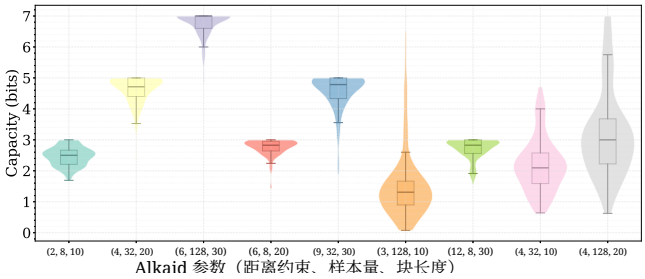
C. 深入分析

Alkaid 的不同参数变体会对系统性能产生不同的影响。为了进一步探究 Alkaid 的各项参数对实验结果的影响，我们使用不同的配置参数进行了受控实验，相关结果分别如图 6 和表 III所示。

距离约束 $d_{\mathcal{T}}$ 是决定系统安全性和鲁棒性的核心变量。图 1 中的数据清楚地表明，增大 $d_{\mathcal{T}}$ 会直接且显著地提高码本中所选码字与其他组的候选码字之间的最小编辑距离，记作 $d_{\min}$。例如，在数值大致相当的配置下



（a）编辑距离分布。（$d_{min} > d_{\mathcal{T}}$，下限约束）



（b）容量分布（比特）。（$C \leq \log_2 k$，上限约束）

图 6. 不同配置参数（$d_{\mathcal{T}}$、 k、 n_l）下鲁棒性与容量之间的权衡

.

表III 不同配置参数（$d_{\mathcal{T}}$、 k、 n_l）下 ALKAID 的性能。

参数			效率			容量		可用性	
$d_{\mathcal{T}}$	k	n_l	编码↑ (令牌/秒)	编码↑ (比特/秒)	解码↑ (比特/秒)	有效载荷↑ (比特/词元)	熵↑ 利用率%	熵	PPL↓
2	8	10	49.3844	11.9794	11.9696	0.2427	22.29%	1.1353	2.4600
4	32	20	32.8736	7.3679	7.3433	0.2249	20.37%	1.1577	2.4686
6	128	30	10.1460	2.1205	2.1085	0.2095	19.11%	1.1686	2.4337
6	8	20	47.6695	6.4399	6.4320	0.1351	12.04%	1.1728	2.4893
9	32	30	30.3108	4.4874	4.4686	0.1483	13.25%	1.1592	2.3825
3	128	10	17.4250	2.1644	2.1550	0.1311	11.18%	1.0837	2.5654
12	8	30	46.9241	4.1630	4.1562	0.0887	7.83%	1.1882	2.5934
4	32	10	36.7648	7.7365	7.7181	0.2125	18.30%	1.1294	2.4358
4	128	20	11.5783	1.6014	1.5942	0.1411	12.68%	1.0793	2.3259

表 IV 优化策略对 Alkaid 性能的影响。

方法	效率				可用性		
	编码↑ (令牌/秒)	编码↑ (比特/秒)	解码↑ (比特/秒)	密度↑ (令牌/秒/GB)	最大 GPU↓ (GB)	熵	PPL↓
None	1.9892	0.3978	0.3977	0.1385	14.5295	1.1543	2.4762
缓存	2.7197	0.5502	0.5500	0.1894	14.4682	1.1632	2.4185
批处理	33.0656	6.7329	6.7104	2.0434	17.7603	1.1493	2.4458
All	33.0280	6.7253	6.7026	2.0411	17.7603	1.1493	2.4458

表 V 不同批次大小下 Alkaid 的性能。

批次大小	效率				可用性		
	编码↑ (令牌/秒)	编码↑ (比特/秒)	解码↑ (比特/秒)	密度↑ (令牌/秒/GB)	最大 GPU↓ (GB)	熵	PPL↓
16	17.0894	3.4934	3.4873	1.0815	16.7625	1.1712	2.4693
32	24.3070	4.9088	4.8966	1.5098	18.0251	1.1745	2.5060
64	32.5706	6.6482	6.6263	2.0131	17.6177	1.1507	2.4478
128	33.0286	6.7249	6.7025	2.0411	17.7603	1.1493	2.4458
144	33.0268	6.7254	6.7027	2.0410	17.7603	1.1493	2.4458

对于样本量 k 和块长度 n_l，将 $d_{\mathcal{T}}$ 从 2（对应配置 (2,8,10)）提高到 12（对应配置 (12,8,30)），会使 $d_{\min}$ 从 5.5 急剧升至 21.0。另一个对比案例中，当 k 固定为 32、 n_l 固定为 20 时，将 $d_{\mathcal{T}}$ 从 4 增至 6，也就是从配置 (4,32,20) 改为 (6,8,20)，会使 $d_{\min}$ 大幅跃升从 9.0 增加到 18.0。这再次确认，该参数为输出中所需的文本差异确立了一个可量化且单调递增的安全下界。然而，表 3 中全面的性能数据表明，这种鲁棒性的增强伴随着显著且持续的系统开销。在提及的第一个对比案例中

，编码效率从每秒 11.9794 比特骤降至 4.1630 比特，而以每个词元的比特数衡量的有效载荷也从 0.2427 大幅降至 0.0887。在第二个对比案例中，编码效率相应地从每秒 7.3679 比特下降至每秒 6.4399 比特，有效载荷则从每个词元 0.2249 比特降至 0.1351 比特。因此， $d_{\mathcal{T}}$ 代表了性能权衡的起点：提高该参数会直接增强安全性，但不可避免地要以降低编码速度和削弱隐写容量为代价。

样本量 k 是调节系统容量的核心变量。图2所示的容量分布表明，增大 k 旨在扩展候选采样空间，从而在理论上探索更高的信息嵌入上限。以固定参数 $d_{\mathcal{T}} = 4$ 和 $n_l = 20$ 的配置为例，当 k 从32增至128时，最大容量从4.9比特升至6.5比特，验证了其提升容量潜力的设计意图。同样，在 $d_{\mathcal{T}} = 6$和 $n_l = 30$的条件下，对比配置(6,8,20)与(6,128,30)可见，容量分布的第三四分位数从3.2比特增至6.9比特。然而，这一潜力的实现受到安全性要求的限制，并会带来沉重的计算负担。在第一个其他参数固定的对比案例中，中位容量并未随 k 增大而上升，反而从4.9比特降至3.1比特，同时分布也更加分散。这表明，在严格的距离约束下，更大的搜索空间可能会加剧结果不确定性，而非稳定地提升有效载荷。更重要的是，计算成本很高。第一个案例中的编码效率从每秒7.3679比特骤降至每秒1.6014比特。第二个对比案例的效率也从每秒6.4399比特显著降至每秒2.1205比特。因此， k 是一个需要仔细权衡的参数：它有望提高容量上限，但其实际收益受到安全约束的严格限制，且计算资源消耗会呈指数级增长。

块长度 n_l 是一个参数，会同时影响鲁棒性边界、容量基线和计算效率。从理论上讲， n_l 定义了每次编码操作所对应的基本文本单元长度，直接影响每次操作可承载信息量的理论上限。数据表明，在其他参数保持不变时，增大 n_l 可以支持更高的有效载荷，不过当它与较高的 $d_{\mathcal{T}}$ 结合时，这一趋势可能逆转。例如，比较配置 (6,8,20) 与 (12,8,30)，尽管 $d_{\mathcal{T}}$大幅增加，且 n_l 从 20 增至 30，有效载荷仍进一步从每个词元 0.1351 比特降至每个词元 0.0887 比特。然而，在 $d_{\mathcal{T}}$和 k相对适中的设置下，例如 $d_{\mathcal{T}} = 4$ 和 $k = 32$，将 n_l 从 10 增至 20，即从配置 (4,32,10) 调整为 (4,32,20)，有效载荷会从每个词元 0.2125 比特略微升至每个词元 0.2249 比特，同时最小编辑距离 $d_{\min}$ 从 4.0 显著增至 9.0。与此同时，增大 n_l 会以线性或非线性的方式扩展编码过程的搜索维度，从而导致计算复杂度显著上升。编码速度（以每秒词元数计）的变化趋势也印证了这一点：在一组比较中，编码速度从 36.7648 降至 32.8736；在另一组比较中，则从 47.6695 降至 46.9241，且 n_l 不断增大。此外， n_l 决定了距离约束 $d_{\mathcal{T}}$所适用的文本范围。较大的 n_l 意味着发散要求会在更长的连续文本片段上实施，因此能够提高对集中错误的鲁棒性。

Alkaid 采用的优化策略显著提升了其效率。 不同优化策略对性能的影响详见表 IV。具体而言，在未应用任何优化时，系统的编码效率仅为每秒 1.9892 个词元，或每秒 0.3978 比特。引入缓存策略后，性能初步提升，编码速度提高至每秒 2.7197 个词元，增幅约为 37%。然而，效率的真正飞跃源于批处理策略的应用。启用批处理后，编码速度飙升至每秒 33.0656 个词元，相较基线版本提升了 16 倍以上。以比特每秒计的有效编码速度也达到 6.7329，呈现出同样显著的提升。从资源利用的角度来看，效率提升并非以资源消耗线性增长为代价。以每 GB GPU 显存每秒处理的词元数衡量的计算密度，从未优化时的 0.1385 大幅提升至启用批处理后的 2.0434。这意味着单位 GPU 显存的编码能力提升了近 15 倍，凸显了这些优化策略在提升硬件利用效率方面的卓越效果。与此同时，系统的可用性指标保持稳定。尽管批处理需要更多显存来执行并行操作，使 GPU 峰值显存使用量从约 14.5 GB 增至 17.8 GB，但生成文本的质量不受影响。熵值保持一致，困惑度也始终维持在约 2.45 的较低水平，表明文本的自然度和流畅度得以保留。总之，Alkaid 的优化策略，尤其是批处理技术，成功将其编码效率提升了一个数量级以上，同时也提高了硬件利用密度。这种效率的显著提升并未损害生成文本的可用性，彰显了该优化方法的工程效能和实用价值。

不同批次大小下 Alkaid 的性能。表V 详细展示了 Alkaid 在不同批次大小下的性能。数据分析表明，增大批大小可显著提升系统的运行效率；性能提升在达到一定规模后趋于稳定，同时所有配置下的文本生成质量都高度一致。在运行效率方面，批大小从 16 增至 128 时，以每秒词元数衡量的编码速度稳步从 17.0894 提升至 33.0286，增幅达 93%；有效比特编码速度则从每秒 3.4934 比特翻倍至每秒 6.7249 比特。值得注意的是，

性能提升呈现出边际收益递减的趋势。从 16 增至 64 时提升最为显著，此后性能曲线趋于平缓，批大小为 128 和 144 时的结果几乎相同。这表明，在当前硬件和模型配置下，批大小介于 64 至 128 之间时，可能已接近性能饱和点。计算密度指标也相应地从每 GB GPU 显存每秒 1.0815 个词元增至 2.0411 个词元，表明较大的批次能更充分地利用并行计算单元，大幅提高硬件效率。同时，GPU 峰值内存消耗保持相对稳定。在可用性方面，所有配置下的关键指标都表现出色且稳定：文本熵保持在 1.15 至 1.17 的狭窄范围内，困惑度则维持在较低水平，为 2.45 至 2.51。这表明，批处理优化大幅提升吞吐量的同时，并未损害生成文本的语义连贯性或语言质量。总之，调整批大小能让 Alkaid 有效提升编码效率和硬件利用密度，同时不影响文本可用性。

VI. 结论

传统的可证明安全隐写术（PSS）提供了强有力的安全保证，但由于易受常见编辑错误影响，在实践中表现脆弱。为弥合理论与鲁棒性之间的差距，我们提出 Alkaid，这是一种通过距离约束编码将差错弹性直接融入安全编码过程的系统。通过确保不同消息对应的载体序列之间具有最小间隔，Alkaid 能够保证，只要累积编辑错误不超过规定上限，消息就始终可以被唯一解码。我们从形式上证明，这种构造在针对有界失真提供确定性鲁棒性的同时，保留了 PSS 的核心特性——计算不可区分性。实验表明，Alkaid 能够有效抵御现实场景中的编辑，从而以严谨且实用的方式统一可证明安全性与差错弹性。

参考文献

[1] G. J. Simmons，“囚徒问题与阈下信道”，载于密码学进展：*Crypto 83* 会议论文集，第 51–67 页，施普林格，1984 年。[2] T. Aura，“数字通信中的实用隐蔽性”，载于信息隐藏国际研讨会，第 265–278 页，施普林格，1996 年。[3] F. Li、Z. Yu 和 C. Qin，“采用交叉反馈机制的基于GAN的空间图像隐写”，《信号处理》，第 190 卷，108341，2022 年。[4] V. Y. Ramandi、M. Fateh 和 M. Rezvani，“Vidagan：用于图像隐写的自适应 GAN”，*IET* 图像处理，第 18 卷，第 12 期，第 3329–3342 页，2024 年。[5] K. R. Malik、M. Sajid、A. Almogren、T. S. Malik、A. H. Khan、A. Altameem、A. U. Rehman 和 S. Hussen，“大数据时代利用离散余弦变换和 GAN 实现安全数据通信的混合隐写框架”，《科学报告》，第 15 卷，第 1 期，第 19630 页，2025 年。[6] T. Bui、S. Agarwal、N. Yu 和 J. Collomosse，“Rosteals：利用自编码器潜在空间的鲁棒隐写”，载于 *IEEE/CVF* 计算机视觉与模式识别会议论文集，第 933–942 页，2023 年。[7] X. Ge、X. Zhang、M. Sun、Y. Wang、L. Li 和 K. SongGong，“通过深度编码器–解码器网络探索跨域冗余以实现语音隐写”，《信息安全与应用杂志》，第 93 卷，第 104150 页，2025 年。

[8] G. Li、S. Li、M. Li、X. Zhang 和 Z. Qian，《隐写网络的隐写术》，载于《AAAI人工智能会议论文集》，第 37 卷，第 5178–5186 页，2023 年。[9] G. Li、S. Li、Z. Qian 和 X. Zhang，《基于深度生成模型的封面可分离固定神经网络隐写术》，载于《第 32 届 ACM多媒体国际会议论文集》，第 10238–10247 页，2024 年。[10] M. Boroumand、M. Chen 和 J. Fridrich，《用于数字图像隐写分析的深度残差网络》，《IEEE信息取证与安全汇刊》，第 14 卷，第 5 期，第 1181–1193 页，2018 年。[11] K. Cheng、Z. Jin、X. Li 和 K. Wu，《确定性文档交换协议及近乎最优的编辑错误二进制码》，载于《2018 年 IEEE 第 59 届计算机科学基础年会（FOCS）》，第 200–211 页，2018 年。[12] G. Kaptchuk、T. M. Jois、M. Green 和 A. D. Rubin，《Meteor：面向现实分布的密码学安全隐写术》，载于《2021 年 ACM SIGSAC计算机与通信安全会议论文集》，第 1529–1548 页，2021 年。[13] J. Ding、K. Chen、Y. Wang、N. Zhao、W. Zhang 和 N. Yu，《Dis-cop：基于“分布副本”的实践中可证明安全隐写术》，载于《2023 年 IEEE安全与隐私研讨会（SP）》，第 2238–2255 页，IEEE，2023 年。[14] M. Bai、K. Pang、G. Liao、J. Yang 和 Y. Huang，《Shimmer：基于熵收集机制的可证明安全隐写术》，载于《第 34 届 USENIX安全研讨会（USENIX Security 25 5965，2025 5949）》，第 – 页。[15] Y. Wang、G. Pei、K. Chen、J. Ding、C. Pan、W. Pang、D. Hu 和 W. Zhang，《Sparsamp：基于稀疏采样的高效可证明安全隐写术》，载于《第 34 届 USENIX安全研讨会（USENIX Security 25）》，2025 年。[16] G. Liao、J. Yang、W. Shao 和 Y. Huang，《可证明安全隐写术设计框架》，载于《第 34 届 USENIX安全研讨会（USENIX Security 25）》，第 6837–6856 页，2025 年。[17] M. Bai、J. Yang、K. Pang、X. Xu、Z. Yang 和 Y. Huang，《非对称资源场景下可证明鲁棒且安全的隐写术》，载于《2025 年 IEEE安全与隐私研讨会（SP）》，第 1438–1456 页，IEEE，2025 年。[18] R. W. Hamming，《检错码和纠错码》，《贝尔系统技术期刊》，第 29 卷，第 2 期，第 147–160 页，1950 年。[19] V. Lcvenshtcin，《可纠正删除、插入和反转的二进制码》，载于《苏联物理学报告》，第 10 卷，1966 年。[20] T. Brown、B. Mann、N. Ryder、M. Subbiah、J. D. Kaplan、P. Dhariwal、A. Neelakantan、P. Shyam、G. Sastry、A. Askell 等，《语言模型是少样本学习器》，《神经信息处理系统进展》，第 33 卷，第 1877–1901 页，2020 年。[21] Anthropic，《Claude 3模型系列：Opus、Sonnet、Haiku》，[22] Deepseek-AI，《Deepseek-R1 通过强化学习促进大语言模型中的推理》，《自然》，第 645 卷，第 633–638 页，2025 年 9 月。[23] G. Team，《Gemini：一系列能力强大的多模态模型》，arXiv 预印本 arXiv:2312.11805，2025 年。[24] Y. Wang、W. Pang、K. Chen、J. Ding、W. Zhang、D. Hu 和 N. Yu，《打破生成式隐写术三难困境：ANStega 实现最优容量、效率与安全性》，载于《第三十三届网络与分布式系统安全研讨会（NDSS ’26）》，互联网协会，2026 年 2 月。[25] S. Zhang、Z. Yang、J. Yang 和 Y. Huang，《可证明安全的生成式语言隐写术》，载于《计算语言学协会论文集：ACL-IJCNLP 2021》（C. Zong、F. Xia、W. Li 和 R. Navigli 编），（线上），第 3046–3055 页，计算语言学协会，2021 年 8 月。[26] C. Cachin，《隐写术的信息论模型》，载于《信息隐藏国际研讨会》，第 306–318 页，施普林格，1998 年。[27] S. Kullback 和 R. A. Leibler，《论信息与充分性》，《数理统计年刊》，第 22 卷，第 1 期，第 79–86 页，1951 年。[28] N. J. Hopper、J. Langford 和 L. Von Ahn，《可证明安全隐写术》，载于《国际密码学年会》，第 77–92 页，施普林格，2002 年。[29] L. Von Ahn 和 N. J. Hopper，《公钥隐写术》，载于《密码学技术理论与应用国际会议》，第 323–341 页，施普林格，2004 年。[30] M. Backes 和 C. Cachin，《应对主动攻击的公钥隐写术》，载于《密码学理论会议》，第 210–226 页，施普林格，2005 年。[31] K. Zeng、K. Chen、W. Zhang、Y. Wang 和 N. Yu，《通过最小化信道错误改进鲁棒自适应隐写术》，《信号处理》，第 195 卷，文章编号 108498，2022 年。

[32] T. Qiao、S. Wang、X. Luo 和 Z. Zhu，《通过改进载体元素选择实现抵抗 JPEG压缩的鲁棒隐写术》，《信号处理》，第 183 卷，文章编号 108048，2021 年。[33] J. Yu、X. Zhang、Y. Xu 和 J. Zhang，《Cross：扩散模型实现可控、鲁棒且安全的图像隐写》，《神经信息处理系统进展》第 36 卷，第 80730–80743 页，2023 年。[34] J. Yang、M. Bai、K. Pang、Y. Gao、J. Zou 和 Y. Huang，《一种新颖的基于语义的文本隐写框架》，《IEEE可信与安全计算汇刊》，2025 年。[35] O. Zamir，《面向语言模型的不可检测隐写术》，《机器学习研究汇刊》，2024 年。[36] M. Christ 和 S. Gunn，《伪随机纠错码》，载于《国际密码学年会》，第 325–347 页，施普林格，2024 年。[37] Y. Qi、N. Zhao、Q. Yao、B. Wu、W. Zhang、N. Yu 和 K. Chen，《STEAD：基于扩散语言模型的鲁棒可证明安全语言隐写术》，载于《第三十九届神经信息处理系统年会》，2025 年。[38] W. Kwon、Z. Li、S. Zhuang、Y. Sheng、L. Zheng、C. H. Yu、J. E. Gonzalez、H. Zhang 和 I. Stoica，《使用 PagedAttention 高效管理大语言模型服务的内存》，2023 年。[39] Qwen，《Qwen2 15115，2025 5 2412 . 技术报告》，arXiv 预印本 arXiv: . 。[40] L. Team，《Llama 3模型群》，arXiv 预印本 arXiv:2407.21783，2024 年。[41] G.-. Team，《GLM-4.5：智能体、推理与编程（ARC）基础模型》，arXiv 预印本 arXiv:2508.06471，2025 年。[42] A. Q. Jiang、A. Sablayrolles、A. Mensch、C. Bamford、D. S. Chaplot、D. de las Casas、F. Bressand、G. Lengyel、G. Lample、L. Saulnier、L. R. Lavaud、M.-A. Lachaux、P. Stock、T. L. Scao、T. Lavril、T. Wang、T. Lacroix 和 W. E. Sayed，《Mistral 7B》，arXiv 预印本 arXiv:2310.06825，2023 年。[43] A. Holtzman、J. Buys、L. Du、M. Forbes 和 Y. Choi，《神经文本退化的奇特案例》，arXiv 预印本 arXiv:1904.09751，2019 年。[44] J. Ye、Z. Xie、L. Zheng、J. Gao、Z. Wu、X. Jiang、Z. Li 和 L. Kong，《Dream 7B：扩散大语言模型》，arXiv 预印本 arXiv:2508.15487，2025 年。[45] W. Qu、W. Zheng、T. Tao、D. Yin、Y. Jiang、Z. Tian、W. Zou、J. Jia 和 J. Zhang，《面向人工智能生成文本的可证明鲁棒多比特水印》，载于《第 34 届 USENIX安全研讨会（USENIX Security 25）》，第 201–220 页，2025 年。[46] J. Kirchenbauer、J. Geiping、Y. Wen、J. Katz、I. Miers 和 T. Goldstein，《面向大语言模型的水印》，载于《国际机器学习会议》，第 17061–17084 页，PMLR，2023 年。[47] Z. Yang、Y. Huang 和 Y.-J. Zhang，《一种快速高效的文本隐写分析方法》，《IEEE信号处理快报》，第 26 卷，631，2019 4，627第 – 期。[48] H. Yang、Y. Bao、Z. Yang、S. Liu、Y. Huang 和 S. Jiao，《通过具有特征金字塔的密集连接长短期记忆网络进行语言隐写分析》，载于《2020 年 ACM信息隐藏与多媒体安全研讨会论文集》，第 5–10 页，2020 年。[49] J. Zou、Z. Yang、S. Zhang、S. u. Rehman 和 Y. Huang，《高性能语言隐写分析、容量估计与隐写定位》，载于《数字水印国际研讨会》，第 80–93 页，施普林格，2020 年。[50] Y. Song、P. Dhariwal、M. Chen 和 I. Sutskever，《一致性模型》，载于《第 40 届国际机器学习会议论文集》，第 32211–32252 页，2023 年。[51] T. M. Jois、G. Beck 和 G. Kaptchuk，《Pulsar：面向扩散模型的安全隐写术》，载于《2024 年 ACM SIGSAC计算机与通信安全会议论文集》，第 4703–4717 页，2024 年。[52] D. Z. Lee、H. Fang、H. Wang 和 E.-C. Chang，《针对人工智能生成内容基于潜变量的水印的移除攻击与防御》，载于《2025 年 ACM SIGSAC计算机与通信安全会议论文集》，第 2174–2188 页，2025 年。[53] Y. Peng、D. Hu、Y. Wang、K. Chen、G. Pei 和 W. Zhang，《Stegad-dpm：基于去噪扩散概率模型的生成式图像隐写》，载于《第 31 届 ACM多媒体国际会议论文集》，第 7143–7151 页，2023 年。

附录 A 证明

A. 定理1的证明

证明。设 M 为随机消息，其条件分布为 $\Pr[M = m \mid \xi] = \frac{|\mathcal{O}_m^\xi|}{k}$。对于任意 $\mathbf{x} \in \mathcal{X}^n$，根据全期望定律，我们有

$$\mathcal{P}_\Pi(\mathbf{x}) = \Pr[\mathsf{Enc}(M,\xi) = \mathbf{x}] = \mathbb{E}_\xi\Big[\Pr[\mathsf{Enc}(M,\xi) = \mathbf{x} \mid \xi]\Big]. \tag{12}$$

给定 ξ，码本 $\mathcal{C} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}\}$ 中的候选序列根据 $\mathcal{P}_\theta$ 独立同分布。定义指示变量 $\mathbf{1}_i(\mathbf{x}) = \mathbf{1}\{\mathbf{x}^{(i)} = \mathbf{x}\}$，则

$$t_\mathbf{x}(\xi) = \sum_{i=1}^{k} \mathbf{1}_i(\mathbf{x}), \tag{13}$$

$$\mathbb{E}[t_\mathbf{x}(\xi)] = \sum_{i=1}^{k} \Pr[\mathbf{x}^{(i)} = \mathbf{x}] = k \cdot \mathcal{P}_\theta(\mathbf{x}). \tag{14}$$

分组 $\{\mathcal{O}_m^\xi\}$ 由 $\mathcal{C}^\xi$ 决定。当随机抽取 η 时，编码器满足：给定 $M = m$，$\mathsf{Enc}(M,\xi)$ 按照规定的概率分布从 $\mathcal{O}_m^\xi$ 中选择一个序列实例。因此，

$$\Pr[\mathsf{Enc}(M,\xi) = \mathbf{x} \mid \xi] = \sum_{m \in \mathcal{M}^\xi} \Pr[M = m \mid \xi] \cdot \Pr[\mathsf{Enc}(M,\xi) = \mathbf{x} \mid M = m,\, \xi]. \tag{15}$$

由此可得

$$\begin{aligned}\Pr[\mathsf{Enc}(M,\xi) = \mathbf{x} \mid \xi] &= \sum_{m:\mathbf{x} \in \mathcal{O}_m^\xi} \frac{|\mathcal{O}_m^\xi|}{k} \cdot \frac{t_\mathbf{x}(\xi)}{|\mathcal{O}_m^\xi|} \\ &= \sum_{m:\mathbf{x} \in \mathcal{O}_m^\xi} \frac{t_\mathbf{x}(\xi)}{k}.\end{aligned} \tag{16}$$

对于固定的 ξ，根据分组策略，$\mathbf{x}$ 属于唯一的组 $\mathcal{O}_{m^*}^\xi$，因此求和可简化为单独一项：

$$\Pr[\mathsf{Enc}(M,\xi) = \mathbf{x} \mid \xi] = \frac{t_\mathbf{x}(\xi)}{k}. \tag{17}$$

对 ξ 取期望，得到

$$\begin{aligned}\mathcal{P}_\Pi(\mathbf{x}) = \mathbb{E}_\xi\left[\frac{t_\mathbf{x}(\xi)}{k}\right] &= \frac{1}{k}\mathbb{E}[t_\mathbf{x}(\xi)] \\ &= \frac{1}{k} \cdot k \cdot \mathcal{P}_\theta(\mathbf{x}) = \mathcal{P}_\theta(\mathbf{x}).\end{aligned} \tag{18}$$

由于 $\mathbf{x}$ 是任意的，因此有 $\mathcal{P}_\Pi = \mathcal{P}_\theta$。根据定义 5，该系统是 **完美安全**。 □

B. 定理2的证明

证明。令 $\mathbf{x} = \mathsf{Enc}(m,\ \xi)$ 和 $\tilde{\mathbf{x}} = \mathcal{E}_e(\mathbf{x})$，并定义随机变量 $D = d(\mathbf{x}, \tilde{\mathbf{x}})$，表示信道引入的基本编辑操作总数。根据信道定义，令 $V_i \in \{0,1\}$ 表示第 i 个位置是否经历了基本编辑操作。那么，$\{V_i\}_{i=1}^n$ 相互独立，且满足 $\mathbb{E}[V_i] = e$，因此有

$$D \le \sum_{i=1}^{n} V_i, \quad \mathbb{E}[D] \le en. \tag{19}$$

如果 $D < \frac{d_{\mathcal{T}}}{2}$，那么对于任意 $m' \neq m$，根据编辑距离的三角不等式，

$$\begin{aligned} d(\tilde{\mathbf{x}}, \mathsf{Enc}(m', \xi)) &\geq d(\mathbf{x}, \mathsf{Enc}(m', \xi)) - d(\mathbf{x}, \tilde{\mathbf{x}}) \\ &> d_{\mathcal{T}} - \frac{d_{\mathcal{T}}}{2} = \frac{d_{\mathcal{T}}}{2}. \end{aligned} \tag{20}$$

同时，我们有 $d(\tilde{\mathbf{x}}, \mathsf{Enc}(m, \xi)) \leq d(\tilde{\mathbf{x}}, \mathbf{x}) < \frac{d_{\mathcal{T}}}{2}$，因此，最小距离解码必然会输出 m。所以，只有当 $D \geq \frac{d_{\mathcal{T}}}{2}$ 时才会发生解码错误，由此可得

$$\Pr[\mathsf{Dec}(\tilde{\mathbf{x}}, \xi) \neq m] \leq \Pr\left[D \geq \frac{d_{\mathcal{T}}}{2}\right]. \tag{21}$$

令 $S = \sum_{i=1}^{n} V_i$ 为 n 个取值于 $[0,1]$，且满足 $\mathbb{E}[S] = en$ 的独立随机变量之和。定义 $t = \frac{d_{\mathcal{T}}}{2} - en > 0$. 。根据霍夫丁不等式，我们有

$$\Pr[S - \mathbb{E}[S] \geq t] \leq \exp\left(-\frac{2t^2}{n}\right). \tag{22}$$

由于 $D \leq S$，因此可得

$$\begin{aligned} \Pr\left[D \geq \frac{d_{\mathcal{T}}}{2}\right] &\leq \Pr\left[S \geq \tfrac{d_{\mathcal{T}}}{2}\right] \\ &= \Pr[S - en \geq t] \\ &\leq \exp\left(-\frac{2}{n}\left(\frac{d_{\mathcal{T}}}{2} - en\right)^2\right). \end{aligned} \tag{23}$$

改写后可得

$$\Pr[\mathsf{Dec}(\mathcal{E}_e(\mathsf{Enc}(m, \xi)), \xi) \neq m] \leq \exp\left(-\frac{(d_{\mathcal{T}} - 2en)^2}{2n}\right), \tag{24}$$

这就完成了证明。 □

C. 定理3 的证明

证明。 编码过程中，系统使用伪随机生成器 PRG 从共享种子 sk 派生编码参数 $\xi = (\zeta_1, \ldots, \zeta_k, \eta)$，其中每个分量均由 PRG(sk) 的输出确定性地生成。在实际实现中，我们可以将 PRG 随机生成的序列划分为块，并将其处理为编码参数。

我们定义以下混合实验。

实验 $\mathsf{Exp}_0(\lambda)$：采样一条消息 $m \sim \mathcal{P}_{\mathcal{M}}$ 和一个共享种子 $\mathsf{sk} \sim \{0,1\}^{\lambda}$。编码器从 PRG(sk) 派生所有编码参数 ξ，并运行 $\mathsf{Enc}(m, \xi)$ 生成输出序列 $\mathbf{x}$，然后将其提供给区分器$\mathcal{A}$。

实验 $\mathsf{Exp}_1(\lambda)$：与 Exp_0，相同，但将 ξ 的所有随机分量替换为从各自定义域中独立均匀采样的样本。

根据 PRG 的计算不可区分性，存在一个可忽略函数 $\mathsf{negl}(\cdot)$，使得对于任意概率多项式时间区分器 $\mathcal{A}$，

$$|\Pr[\mathcal{A}(\mathsf{Exp}_0(\lambda)) = 1] - \Pr[\mathcal{A}(\mathsf{Exp}_1(\lambda)) = 1]| \leq \mathsf{negl}(\lambda). \tag{25}$$

接下来，考虑 $\mathsf{Exp}_1(\lambda)$ 所诱导的输出分布。在此实验中，编码参数的各个分量 $\xi = (\zeta_1, \ldots, \zeta_k, \eta)$ 相互独立，并根据各自的定义域独立采样。

具体而言，在码本构造过程中，每个分量ζ_i都可以表示为一个随机向量 $\zeta_i = (r_{i,1}, \ldots, r_{i,n_l})$，其中每个元素$r_{i,\ t}$相互独立且服从均匀分布。因此，$\xi$ 的联合分布满足定理1所要求的所有独立性和分布匹配条件。

附录 B 鲁棒性分析

此外，自适应消息编码机制不会改变消息空间 $\mathcal{M}^{\xi}$ 上的诱导分布，因此 $\mathcal{P}^{\xi}_{\mathcal{M}}(m) = \frac{|\mathcal{O}^{\xi}_{m}|}{k}$.

因此，根据定理1，系统在$\mathsf{Exp}_1(\lambda)$中是完美安全的。因此，我们有 $\mathcal{P}_{\Pi} = \mathcal{P}_{\theta}$. 因此，

$$\Pr[\mathcal{A}(\mathsf{Exp}_1(\lambda)) = 1] = \Pr_{\mathbf{x} \sim \mathcal{P}_{\theta}}[\mathcal{A}(1^{\lambda}, \mathbf{x}) = 1]. \tag{26}$$

将上述不等式与三角不等式结合可得

$$\begin{aligned} \mathsf{Adv}_{\mathcal{A},\Pi}(\lambda) &= \left|\Pr[\mathcal{A}(\mathsf{Exp}_0(\lambda)) = 1] - \Pr_{\mathbf{x} \sim \mathcal{P}_{\theta}}[\mathcal{A}(1^{\lambda}, \mathbf{x}) = 1]\right| \\ &\leq \mathsf{negl}(\lambda). \end{aligned} \tag{27}$$

这表明系统 Π 满足定义6，并且是**计算安全的**。 □

定理2中的鲁棒性证明基于一种最坏情况假设，即码字之间的最小距离恰好等于设计阈值 $d_{\mathcal{T}}$，从而得到相应的理论界。然而，在Alkaid的实际实现中，样本量 k 远小于码本空间，这种稀疏性使得由 $d_{\mathcal{T}}$导出的理论界偏于保守。为在现实配置下验证Alkaid的鲁棒性，我们开展了一项针对性实验：从长度为100的100元字母表中随机采样100个码字，并施加严格约束，使任意一对码字之间的汉明距离均超过 $d_{\mathcal{T}}$，同时将欧几里得距离作为最小距离译码中的硬判决准则。如图8所示，所有沃罗诺伊胞均向外扩展，并完全包络理论汉明球。尽管由于随机采样引起的局部密度波动而呈现出不规则的几何形状，每个胞都严格超出理论边界。这一现象直接表明，稀疏性扩大了实际译码区域，从而使有效纠错半径普遍超过$d_{\mathcal{T}}/2$。结合表II中的观察结果，即 $d_{\min}$ 始终超过 $d_{\mathcal{T}}$，Alkaid 部署后的解码失败率将低于由 $d_{\mathcal{T}}$导出的最坏情况理论上界，从而为其在真实世界噪声环境中的可靠部署提供实质性保障。

此外，由于 Alkaid 采用分块处理架构，纠错能力仅限于各个数据块内部，无法跨数据块提供支撑。因此，该系统抵御集中成簇错误的能力有限



图 7. 在一致性模型上采用距离约束编码的样本。

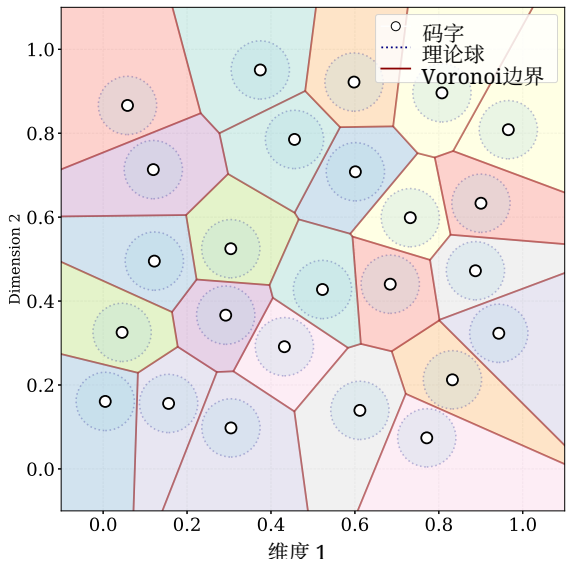


图 8. 实际 Voronoi译码区域。

表 VI Alkaid 的隐写分析结果。

方法	检测率%
FCN [47]	50.11±0.22%
R-BiLSTM-C [48]	49.82±0.43%
LSTMATT [49]	50.05±0.49%

，而这类错误集中在局部区域。因此，实际部署时需要谨慎选择块大小 n_l 和样本量 k，以平衡计算效率与鲁棒性。

附录 C 隐写测试结果

为全面评估 Alkaid 的安全性，我们使用多种成熟的隐写分析方法对其进行了实证测试。目标是区分随机采样文本与 Alkaid 生成的隐写文本。我们采用了三种深度

表 VII 不同生成模型中距离约束编码的性能。

方法	有效载荷	速度	成功率
Dream [44]	0.13 比特/词元	0.11 比特/秒	100%
CM-64x [50]	2.2e-3 比特/像素	5.91 比特/秒	100%
CM-256x [50]	1.2e-3 比特/像素	0.63 比特/秒	100%

已被证明有效的基于学习的方法：FCN [47]，R-BiLSTM-C [48]，和 LSTMATT [49]。

如表 VI所示，Alkaid 的检测率接近 50%。在隐写分析中，低于 0.5% 的偏差通常被视为可忽略不计。因此，我们的实验结果显示，检测率低于 0.5%，这表明当前的隐写分析方法在识别 Alkaid 生成的内容方面并不优于随机猜测。这些发现为 Alkaid 的安全性提供了实证支撑。

附录 D
尝试应用于其他生成模型

尽管以往研究取得了一定成果，但通常依赖于能够提供明确概率分布的生成模型。这一要求限制了它们在扩散模型和变分自编码器等生成方法中的适用性，因为这些模型通常不会直接输出概率分布 [51], [52], [53]。相比之下，距离约束编码只要求模型能够生成样本，因此适用于范围更广的生成模型。为验证这一点，我们对若干生成模型进行了实验评估。

首先，我们采用了 Dream-7B [44]，这一基于去噪机制的扩散语言模型。通过控制其随机采样过程，我们生成了 256 个长度为 64 的序列，采样步数为 300。表VII中的结果表明，与表II中的 STEAD方法相比，距离约束编码的嵌入容量和速率并未显著下降。由于

扩散模型生成过程中的不确定性更高，样本之间的差异也更大，因此解码成功率相应提高。

此外，我们还在一致性模型 [50]上进行了实验。由于该模型无法直接提供概率分布，我们通过调整初始随机噪声来控制生成过程，同时仍能保持安全性。我们测试了两种模型规模：一种生成 256 张尺寸为64×64 的图像，另一种生成 256 张尺寸为 256×256的图像。如图7 和表VII所示，较小模型的嵌入率和速度较高，但在较大模型中有所下降。不过，由于生成图像之间存在显著差异，解码成功率仍然很高。这表明图像本身具有较高的熵，可为编码提供潜在空间。然而，直接进行全局编码的效率较低，因此需要采用类似 Alkaid 的基于分块的机制，将距离约束编码的指数级复杂度降至线性级，从而提升实际效率。

这些实验表明，距离约束编码在不同类型和规模的生成模型上均表现良好。其核心优势在于仅依赖样本生成能力，而不依赖特定的概率分布形式。这为在各种模型中构建统一的距离约束编码框架创造了可能。未来的工作可探索针对不同模型特征的优化策略，例如分块、剪枝和并行化。受篇幅限制，本文主要关注自回归大语言模型中的高效实现。在不同生成范式下进行系统优化和理论完善，仍是值得进一步研究的重要方向。