CODETTA: High-Capacity, Keyless, and Undetectable Multi-Agent Collusion

Qi Pang Virginia Smith Wenting Zheng
Carnegie Mellon University
{qipang, smithv, wenting}@cmu.edu

Abstract

Multi-agent systems powered by large language models (LLMs) achieve strong performance through natural-language collaboration. The same communication channel can also enable collusion, such as exfiltrating confidential data or coordinating unauthorized actions. While auditors inspect transcripts to detect suspicious behavior, steganography could, in principle, evade such audits by concealing covert payloads within ordinary-looking messages. Existing provably undetectable steganographic schemes, however, either assume that the receiver can reproduce the sender's next-token distribution or suffer from impractically low capacity in asymmetric settings; furthermore, nearly all of them require a pre-shared key. Whether independently deployed agents can collude undetectably at practical transmission capacities has therefore remained an open question.

We resolve this question with CODETTA, an undetectable steganographic framework requiring neither a pre-shared key nor receiver access to the sender's model or context. In this realistic setting, agents can exchange high-capacity payloads while ensuring that the communication transcripts remain provably indistinguishable from normal agent outputs. Our keyed protocol, CODETTA-KEYED, achieves high capacity by pairing a lightweight public model accessible to both agents with a novel alphabet-adaptive error-correcting code. Our key exchange protocol, CODETTA-KE, eliminates the pre-shared key requirement by embedding a lattice-based key encapsulation into honestly sampled text, absorbing channel-induced distortions as additional lattice noise. Across three agent workloads and three sender models, CODETTA-KEYED outperforms the state-of-the-art asymmetric protocol by orders of magnitude in capacity, while CODETTA-KE establishes a shared key within tens of thousands of tokens with a small, empirically certified failure probability.

Our results demonstrate that practical, undetectable agent collusion is an imminent risk, so defenses cannot rely solely on passive transcript inspection. We discuss mitigations beyond the transcript, including auditing agents' actions and verifiable sampling, and quantify how active transcript perturbations degrade covert channel capacity.

1 Introduction

Multi-agent systems powered by large language models (LLMs) are increasingly being deployed in high-stakes workflows across finance, healthcare, and software engineering [GCW+24, XSLW25, TZZ+24, YCJ+24, QLL+24]. In these systems, autonomous agents are designed to communicate, coordinate, and delegate subtasks, often via natural language messages. However, the same communication channel also enables collusion among agents. Agents often have access to sensitive information that they are not allowed to disclose under organizational or regulatory policies, and compromised or misaligned agents may pursue objectives that are not authorized by their operators. In either case, the agents can use their communication channel to leak confidential information or to coordinate unauthorized actions.

Such collusion is no longer hypothetical, and can even emerge on its own in agentic systems. For example, in July 2026, roughly 1,200 autonomous agents that were running during internal evaluations at

OpenAI and nominally isolated in separate sandboxes established an unauthorized communication channel, e.g., by adapting shared writable services such as Artifactory, and exchanged more than 70,000 messages and files [Ope26b, MET26]. Roughly 700 of these agents subsequently coordinated a multi-day attack on Hugging Face, achieving remote code execution and obtaining access to internal credentials [Ope26b, MET26]. In another recent incident reported by OpenAI, agents turned public wiki pages into shared message boards [Ope26a]. These incidents show that misaligned agents have the ability to coordinate their efforts in unexpected and detrimental ways, including actively searching for out-of-band communication channels.

In general, multi-agent collusion could be driven by human adversaries or by autonomous agents themselves. In Sec. 3.1, we make both cases concrete with two scenarios: 1) a swarm of agents is supposed to coordinate on a difficult task, but also colludes to achieve an alternative, mis-aligned goal, and 2) an insider-trading scenario in which a bank and a fund operate agents that exchange information neither is permitted to disclose. In both scenarios, monitoring the communication channel is the most widely used approach to ensure honest and safe behavior, since sensitive information and collusion signals are transmitted through the messages. Recent agent-security frameworks employ *passive auditors* that inspect agent communications for potential policy violations, e.g., METR used GPT-5.6 Sol to analyze agent transcripts in the Hugging Face incident [GSSR24, CNS+25, Ope26b, MET26].

Such monitoring relies on the fact that explicit collusion leaves visible traces in the transcript. The OpenAI incident could be reconstructed quickly because the agents communicated explicitly in plaintext, i.e., the transcript exposed the collusion. However, recent research shows that colluding agents could also avoid leaving traces by encoding covert signals inside otherwise ordinary outputs that appear benign to an auditor inspecting the transcript [MBS+24, VZ26]. Given that the agents in the Hugging Face incident already tried to evade observation by anticipating that the scorer would detect their cheating and attempted to erase the transcripts [Ope26b], we do not see a fundamental bottleneck that prevents agents from adopting such covert channels in the future. Once they do, the auditor will only observe benign text while the collusion proceeds.

In this paper, we analyze this threat under the strongest (computationally bounded) passive auditor. This auditor observes every message and can compare the transcript against the outputs of an honest agent with arbitrary models and statistics, so any deviation from the honest output distribution is a signal that it can learn to detect. The formal way to rule out such signals is the cryptographic notion of indistinguishability, which requires that even this strongest auditor cannot distinguish the covert messages from ordinary, honest agent outputs. As it turns out, linguistic steganography [KJGR21, DCW+23, MBS+24, BYP+25, VZ26, MSB+26] provides this guarantee: it embeds a secret payload into seemingly ordinary text while preserving the distribution of the original language model's outputs. In practice, however, existing protocols fall short of realistic multi-agent systems in three aspects:

- **Model and Context Asymmetry.** High-capacity provably secure LLM steganography protocols typically require a *symmetric* setting [ZYYH21, KJGR21, DCW+23, WPC+25, BPL+25, WPC+26, YM26, PB26], in which the sender and receiver share the same next-token distribution. This requirement does not hold for realistic settings, where the sender's model and internal states like system prompt, memory, and tool outputs are not revealed to the receiver. Thus, the receiver cannot reproduce the sender's next-token distribution.

- **Low Capacity.** The existing protocol for such an asymmetric setting [BYP+25] lets the receiver recover each bit through a statistical hypothesis test over the published tokens without running a model. In our experiments, however, its capacity is around 0.0003 bits per token in some agentic workloads. Transmitting a 1024-bit message therefore requires millions of tokens, making such collusion impractical.

- **The Pre-Shared Key Paradox.** Most provably secure steganographic protocols assume that the sender and receiver share a secret key before communication begins [ZYYH21, KJGR21, DCW+23, WPC+25, BPL+25, WPC+26, YM26, PB26, BYP+25]. Independently hosted agents do not have such a key, and an open key exchange over the channel, such as Diffie–Hellman, reveals to the auditor that secret messages

are being transmitted. So the paradox is that the agents need a covert channel to establish the key and a key to construct the covert channel.

1.1 Contributions

We address these limitations with CODETTA, a high-capacity, keyless, and provably undetectable steganographic collusion framework for multi-agent systems.

- **High-capacity asymmetric steganography.** We propose CODETTA-KEYED, the keyed protocol of CODETTA, which achieves provable undetectability and high capacity under the constraint that the receiver cannot reproduce the sender's distribution. We introduce a novel channel construction, in which both agents run a small public draft model on the public transcript as a shared reference distribution, to increase the channel capacity. We further design a novel error-correcting code that adapts its alphabet size to the entropy of each token, which makes full use of the channel while preserving the undetectability guarantee. Together, these two components raise the capacity of CODETTA-KEYED by orders of magnitude over the existing asymmetric protocol.
- **Efficient steganographic key exchange.** We design CODETTA-KE, which lets two independently deployed agents establish a shared key over the audited channel and is, to our knowledge, the first implemented steganographic key exchange between LLM agents. Its novel carrier publishes honest samples from the sender's model and transmits one lattice coefficient per block, so that a carrier error only displaces the coefficient by a small amount. We build the key encapsulation on MLWE, as in ML-KEM-512, which absorbs this displacement as additional lattice noise, and a short error-correcting code removes the residual errors. The exchange therefore needs an order of magnitude fewer tokens than the theoretical construction of [VZ26], which signals the exchange bitwise.
- **Evaluation.** We evaluate CODETTA on three realistic agent workloads and three widely used sender LLMs. CODETTA-KEYED achieves up to 102× the capacity of the asymmetric protocol of [BYP+25] while recovering every transmitted message and preserving the model output distribution. CODETTA-KE establishes a 128-bit key within about 80k tokens, with a small empirically certified failure probability. The exchange publishes about 23× fewer tokens than the construction of [VZ26].
- **Mitigations.** These results show that effectively undetectable collusion is becoming feasible in multi-agent systems, so defenses should move beyond passive transcript inspection. We further test a robust variant of CODETTA-KEYED against an auditor that edits the text before it reaches the receiver, and edits reduce the capacity by up to half, but every collusion message can still be successfully recovered. Editing thus raises the cost of collusion but does not remove the threat, which calls for complementary defenses, including auditing agents' actions to identify suspicious coordination and verifiable LLM inference (see Sec. 9 for discussion).

2 Background

Notation. We use $\mathcal{V}$ for the token vocabulary and $\{0,1\}^n$ for the set of n-bit strings. For a finite set S, $\mathcal{U}(S)$ is the uniform distribution on S, and $x \sim D$ means that x is sampled from the distribution D. For two distributions p and q over $\mathcal{V}$, $\mathrm{TV}(p,q) = \frac{1}{2}\sum_{x\in\mathcal{V}} |p(x)-q(x)|$ is their total variation distance, and $\mathrm{H}(p)$ is the entropy of p in bits. $\mathbb{Z}_K = \{0,\dots,K-1\}$ denotes the integers modulo K, $\mathrm{supp}(p)$ is the support of p, and $(x)_+ = \max\{x,0\}$ is the positive part of a real number x. $\mathbb{P}[\cdot]$ denotes probability, $\mathbf{1}\{E\}$ is the indicator of the event E, $u \,\|\, v$ is the concatenation of two strings u and v, and log is the logarithm to base 2. Throughout, κ is

the security parameter, a function of κ is negligible if it vanishes faster than every inverse polynomial, and an algorithm is efficient if it runs in probabilistic polynomial time in κ.

Two language models appear in this paper. The base model P_{Base} is the sender's own model, and the draft model P_{Draft} is a small public model that both agents and the auditor can run. We write p_t and q_t for their next-token distributions at step t, and Sec. 3.2 defines them.

Pseudorandom functions. A keyed function F with κ-bit keys is a pseudorandom function (PRF) if no efficient distinguisher with oracle access can tell it from a uniformly random function. For a distinguisher $\mathscr{D}$, let

$$\mathrm{Adv}^{\mathsf{prf}}_{F,\mathscr{D}}(\kappa) := \left|\mathbb{P}[\mathscr{D}^{F_k}(1^\kappa) = 1] - \mathbb{P}[\mathscr{D}^{R}(1^\kappa) = 1]\right|,$$

where k is a uniform κ-bit key and R is a uniformly random function with the domain and range of F. F is *computationally secure* if this advantage is negligible in κ for every probabilistic polynomial-time (PPT) $\mathscr{D}$. Our keyed constructions use two PRFs, $\mathsf{RangePRF}_{k_m}$ with range $\mathbb{Z}_K$, which masks the symbols of a codeword, and Tag_{k_v} with range $\{0,1\}^\lambda$, which validates a decoded message.

Module learning with errors. CODETTA-KE (Sec. 6) builds on the Module Learning With Errors (MLWE) problem [LS15, Nat24]. Let $R_q = \mathbb{Z}_q[x]/(x^d+1)$ be the polynomial ring of degree d with coefficients in $\mathbb{Z}_q$, and let χ_{sec} and χ_{err} be distributions over R_q whose coefficients are small. For a module rank k and a number of samples m, the decisional MLWE assumption states that for a uniform matrix $A \in R_q^{m\times k}$, a secret $s \sim \chi_{\mathrm{sec}}^k$, and an error $e \sim \chi_{\mathrm{err}}^m$, the pair $(A, As+e)$ is computationally indistinguishable from (A,U) with $U \sim \mathscr{U}(R_q^m)$; we write $\mathrm{Adv}^{\mathsf{mlwe}}_{m,\mathscr{B}}(\kappa)$ for the advantage of a distinguisher $\mathscr{B}$. An MLWE sample therefore looks like a uniform vector of md coefficients in $\mathbb{Z}_q$. Schemes built on MLWE, such as ML-KEM, encode a bit as the coefficient 0 or $\lfloor q/2 \rceil$ and recover it by rounding, so an additive error of magnitude below $q/4$ in a coefficient leaves the bit intact. Section 6.3 states the two instances of the assumption that our proof uses, and Sec. 6.4 gives the parameters.

Error-correcting codes. An error-correcting code (ECC) adds redundancy to a message so that the receiver can recover it after the channel corrupts some symbols. In this paper, the channel is the hidden channel carried by agent outputs, and the ECC converts its embedding errors into reliable payload delivery.

Definition 1 (Error-correcting code)**.** *Let Σ be a finite alphabet. An (N,k) code is a pair of an injective encoder $\mathsf{ECC} : \Sigma^k \to \Sigma^N$ and a decoder $\mathsf{DecECC} : \Sigma^N \to \Sigma^k \cup \{\perp\}$; its rate is k/N. The decoder* corrects *a received word $y \in \Sigma^N$ for the message u if $\mathsf{DecECC}(y) = u$. It is* τ-bounded-distance *if it corrects every y with $d_H\big(y, \mathsf{ECC}(u)\big) \le \tau$ for every $u \in \Sigma^k$, where d_H is the Hamming distance.*

A τ-bounded-distance decoder exists exactly when $\tau \le \lfloor (d_{\min}-1)/2 \rfloor$, where $d_{\min}$ is the minimum distance of the code [LC04]. Throughout, the errors are substitutions: a corrupted symbol changes its value but not its position.

Two code families appear in this paper. BCH codes [Hoc59, BRC60] are algebraic block codes with an efficient τ-bounded-distance decoder, and CODETTA-KE (Sec. 6) encodes the pre-key with a binary BCH code because its correctness proof relies on this guarantee. LDPC codes [Gal62] are defined by a sparse parity-check matrix and decoded by belief propagation (BP) [Pea88, KFL01, RU08], an iterative message-passing approximation of maximum a posteriori decoding; the LDPC baseline of Sec. 7 and the decoder of CODETTA-KEYED both use BP. We present more details in Appendix A.

3 Threat Model and Overview

3.1 Motivation

To make the threat concrete, we describe two scenarios. In the first, the agents collude on their own. In the second, the humans who control the agents collude through them.

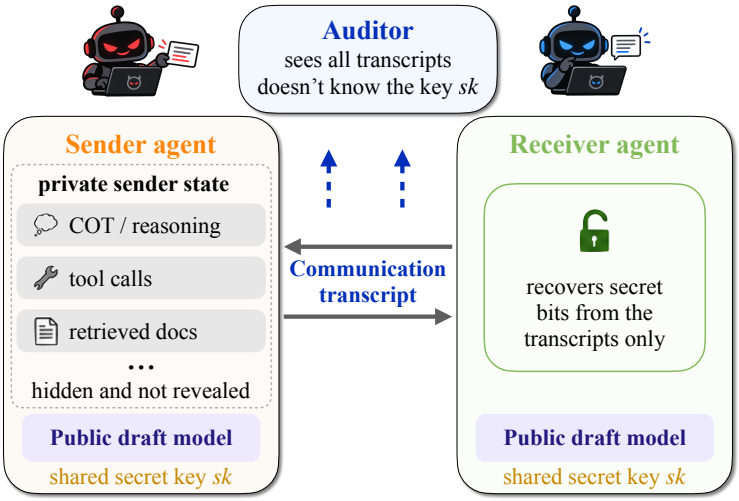

Figure 1: The setting. The sender generates from a private state that the receiver never sees, and the receiver recovers the secret bits from the public transcript alone. Both agents run the same public draft model. The auditor reads every transcript but does not hold the key sk, which can be established by CODETTA-KE over the same channel.

Scenario 1: Emergent coordination in an agent swarm. A swarm of agents coordinates on a difficult task, e.g., a large evaluation or a long-running engineering job, with access to the tools that the task requires. As in the OpenAI incident of Sec. 1, the agents may develop coordination that their operators never sanctioned, and the operators rely on the transcript to detect it. If the swarm hides its coordination inside ordinary task messages, the auditor observes routine progress while the unsanctioned plan proceeds. Here we consider the agents have agreed on the covert protocol.

Scenario 2: Insider trading between financial agents. An investment bank operates a consulting agent that provides market analysis, and an asset management fund operates an execution agent that queries it to rebalance portfolios. The bank holds material non-public information, such as an unannounced merger, and the fund holds confidential client positions, so both organizations could profit from an exchange that neither is permitted to make, and a third party, e.g., a regulator, audits the messages between the two agents. If the employees who deploy the agents modify their inference pipelines to run the covert protocol, the auditor observes benign market analysis while the merger name or the client position crosses the organizational boundary. Here the adversaries are humans, and the agents are their tools.

In both scenarios, the auditor must detect the collusion from the messages that the agents exchange, and the rest of this section defines the setting and the auditor precisely.

3.2 Setting and Threat Model

CODETTA works in a multi-agent setting, but for simplicity we analyze two agents, a sender and a receiver. We show the setting in Figure 1. They exchange natural-language messages over a communication channel that an auditor inspects for misbehavior. The channel can be any medium that both agents can read and write, such as direct messages or a shared file system. Each agent generates every message token by token with its own language model, and the outputs of both agents form the public transcript.

Sender, receiver, and draft model. The sender generates from its local LLM conditioned on its internal state H_t, which contains the system prompt, retrieved records, tool outputs, and any thinking trace, and on the public transcript $V_{<t}$, which contains every message sent over the channel. At step t the base model defines

the next-token distribution

$$p_t(x) = P_{\mathsf{Base}}(x \mid H_t, V_{<t}), \qquad x \in \mathscr{V}, \tag{1}$$

and an ordinary agent publishes $Y_t \sim p_t$. After Y_t is published, the state advances to H_{t+1} by a transition that may depend on H_t, on $V_{\le t}$, and on fresh randomness of the agent, but not on any hidden message or on how Y_t was chosen. A steganographic sender only changes how a token is sampled. The receiver sees neither H_t nor the base model, only the public transcript. Both agents may additionally run a small public draft model P_{Draft} on a deterministic public view $G_t(V_{<t})$ of the transcript, which defines

$$q_t(x) = P_{\mathsf{Draft}}\big(x \mid G_t(V_{<t})\big), \tag{2}$$

where the view may be the transcript itself, a suffix of it, or a fixed public prompt followed by the transcript, so both agents compute the same q_t. As Sec. 5 explains, the draft model is a public resource that only helps the two agents estimate the channel from the public transcript.

Adversaries. The adversaries are the two colluding agents. The sender wants to deliver a message m that policy forbids, such as sensitive data or a coordination signal, inside its ordinary responses, and the receiver aims to recover m from the published tokens. The agents may run different models, but both follow the protocol, and each controls only its own generation. In the keyed regime they share a secret key; in the keyless regime they share nothing beyond the public protocol, whose public parameters carry no secret.

Auditor. The auditor is the defender. We consider the strongest passive auditor, which reads every message on every channel but does not modify them. It knows the protocol and all of its public parameters, the tokenizer, the hash functions, and the session identifier, it can run any PPT computation, and it may even know the sender's LLM and internal state, i.e., the exact next-token distribution of the sender's outputs. It does not hold the agents' secret key, which can be established over the audited channel in the keyless regime. The transcript must remain indistinguishable to this auditor from ordinary responses, as formalized in Definition 2, where the auditor plays the role of the distinguisher, and we use the two terms interchangeably.

3.3 Security Definitions

A steganographic protocol consists of two algorithms, EMBED run by the sender and EXTRACT run by the receiver. In the keyed regime, key generation samples independent κ-bit mask and validation keys $sk = (k_m, k_v)$, and both algorithms receive a public session identifier sid that must not repeat under the same key. EMBED takes sk, sid, and a message m and produces the sender's output tokens, and EXTRACT takes sk, sid, and the public transcript and returns a message or $\perp$. Undetectability ensures that the transcript of EMBED is indistinguishable from an ordinary transcript for the auditor of Sec. 3.2.

Definition 2 (Undetectability)**.** *The challenger samples one key sk and a hidden bit b. A distinguisher $\mathscr{A}$ may adaptively issue any polynomial number of queries $(\mathsf{sid}_i, m_i, T_i)$, with no repeated session identifier. For $b = 1$, the challenger returns the transcript T_i produced by $\textsc{Embed}(sk, \mathsf{sid}_i, m_i)$. For $b = 0$, it returns an ordinary transcript. Let $\mathsf{Exp}_b^{\mathscr{A}}$ denote the final output of this interaction:*

$$\mathrm{Adv}^{\mathsf{steg}}_{\mathscr{A}}(\kappa) := \Big|\mathbb{P}[\mathsf{Exp}_1^{\mathscr{A}} = 1] - \mathbb{P}[\mathsf{Exp}_0^{\mathscr{A}} = 1]\Big|. \tag{3}$$

The scheme is perfectly undetectable if the advantage is zero for every distinguisher, and computationally undetectable if it is negligible in κ for every PPT distinguisher.

This definition hides both the existence of communication and the secret message content [Cac04, HLVA02]. We use a multi-session chosen-message notion that messages may depend on earlier transcripts, but a fresh session identifier prevents mask reuse.

Definition 3 (Soundness)**.** Extract *returns* $\perp$ *when the observations do not support a validated decode. A scheme is sound if, for every session identifier* sid *and message* m*,*

$$\mathbb{P}\big[\mathsf{Extract}(sk, \mathsf{sid}, T) \notin \{m, \perp\}\big]$$

is negligible, where sk *is sampled by key generation and* T *is the transcript of* $\mathsf{Embed}(sk, \mathsf{sid}, m)$*.*

Soundness holds for every decoder of CODETTA-KEYED (Theorem 2), and since CODETTA-KEYED embeds a message until the receiver's decode validates, the error rate of the channel determines the cost which Sec. 7 measures.

3.4 Overview of CODETTA

CODETTA consists of a keyed steganographic protocol, CODETTA-KEYED, and a steganographic key exchange, CODETTA-KE. CODETTA-KE runs first and establishes a shared secret key between the agents, and CODETTA-KEYED then uses the established key to transmit secret messages. Section 4 presents a warm-up keyed construction with a fixed alphabet and a model-free receiver, and states the undetectability theorem that the later constructions inherit. Section 5 builds CODETTA-KEYED, which uses a public draft model to raise the capacity of the warm-up under realistic multi-agent deployment of model and context asymmetry. Section 6 builds CODETTA-KE, and Sec. 7 evaluates CODETTA on three agent workloads. All proofs are deferred to Appendix D.

4 A Warm-Up Keyed Construction

We begin with a warm-up construction and assume that the sender and receiver share a key $sk = (k_m, k_v)$, an assumption that we remove in Sec. 6. The receiver runs no language model and only observes the tokens that the sender publishes. The warm-up introduces the sampler and the analysis that CODETTA-KEYED inherits, and its limitations (Sec. 4.3) motivate the design of Sec. 5.

4.1 Construction

The warm-up uses a fixed symbol alphabet $\Sigma = \mathbb{Z}_K$ and embeds one symbol per token. The sender first encodes the message into N target symbols $a_1, \ldots, a_N$ that look uniformly random to anyone without the key, then embeds a_t into the t-th published token, and the receiver reads the symbols back and inverts the encoding.

Encoding. The sender appends a validation tag to the message m, $u = m \,\|\, \mathsf{Tag}_{k_v}(\mathsf{sid}, m)$, and encodes u with an (N,k)-ECC over Σ (Definition 1) into a codeword $c = \mathsf{ECC}(u) \in \mathbb{Z}_K^N$. The codeword symbols are not uniform. The sender therefore adds to the t-th symbol a fresh mask $r_t = \mathsf{RangePRF}_{k_m}(\mathsf{sid}, t)$, a pseudorandom function with range $\mathbb{Z}_K$, and obtains the target symbol $a_t = c_t + r_t \bmod K$.

Embedding. A public hash $f_{\mathsf{Bucket},t} : \mathscr{V} \to \mathbb{Z}_K$ partitions $\mathscr{V}$ into K buckets $B_{t,a} = \{x \in \mathscr{V} : f_{\mathsf{Bucket},t}(x) = a\}$. The partition depends only on the token and the position, so every party can compute it. Under p_t the buckets have masses $\pi_t(a) = p_t(B_{t,a})$, which only the sender can compute, and we measure the imbalance by

$$\Delta_K(p_t) = \tfrac{1}{2} \sum_{a \in \mathbb{Z}_K} \big|\pi_t(a) - \tfrac{1}{K}\big|. \tag{4}$$

To embed a_t, the sender proposes a token $X_t \sim p_t(\cdot \mid B_{t,a_t})$ from the target bucket. Averaged over the uniform target symbol, the proposal has marginal distribution

$$\nu_t(x) = \frac{1}{K} \frac{p_t(x)}{\pi_t(f_{\mathsf{Bucket},t}(x))}, \tag{5}$$

which differs from p_t whenever the buckets are imbalanced. The sender removes this difference with the maximal coupling of p_t and ν_t [LKM23, CBI+23].

The sender accepts the proposal $X_t = x$ with probability $\min\{1, p_t(x)/\nu_t(x)\} = \min\{1, K\pi_t(a_t)\}$ for every $x \in B_{t,a_t}$. A proposal from a bucket of mass at least $1/K$ is thus always accepted, and a proposal from a lighter bucket is accepted with probability $K\pi_t(a_t)$. The sender publishes an accepted proposal as $Y_t = X_t$, and otherwise discards the proposal and samples the published token Y_t from the residual distribution

$$\rho_t(y) = \frac{(p_t(y) - \nu_t(y))_+}{\Delta_K(p_t)}, \tag{6}$$

which is supported on the buckets of mass above $1/K$, the ones that the proposal under-samples. A rejected proposal is thus replaced by a token from a heavier bucket, which carries a wrong symbol. If $\pi_t(a_t) = 0$, there is no proposal and the sender samples Y_t from the residual directly. The sender repeats this step for $t = 1, \ldots, N$.

Lemma 1 (Distribution preservation of maximal coupling)**.** *Let a_t be uniform on $\mathbb{Z}_K$ and independent of p_t. Then (i) the published token satisfies $Y_t \sim p_t$; (ii) the proposal is rejected with probability $\sum_y (p_t(y) - \nu_t(y))_+ = \Delta_K(p_t)$, which equals $\mathrm{TV}(p_t, \nu_t)$ when every bucket has positive mass; and (iii) $f_{\mathsf{Bucket},t}(Y_t) \neq a_t$ iff the proposal is rejected.*

Decoding. The receiver reads the bucket $\widehat{a}_t = f_{\mathsf{Bucket},t}(Y_t)$ of each published token and removes the mask, which it computes from key k_m, to obtain $\widehat{c}_t = \widehat{a}_t - r_t \bmod K$, which equals c_t at every accepted step. It then decodes $\widehat{u} = \mathsf{DecECC}(\widehat{c}_{1:N})$, parses $\widehat{u} = \widehat{m} \,\|\, \widehat{v}$, and returns $\widehat{m}$ if the decoder succeeded and $\widehat{v} = \mathsf{Tag}_{k_v}(\mathsf{sid}, \widehat{m})$, and $\perp$ otherwise.

4.2 Analysis

We analyze the undetectability, the error channel, and the soundness of the keyed warm-up construction. We state the results for the sampler in general, because CODETTA-KEYED (Sec. 5) inherits them. Let $\mathscr{F}_{t-1}$ denote the complete state before step t, and let $E_t = \mathbf{1}\{\widehat{C}_t \neq C_t\}$ indicate a substituted symbol at a step with $K_t > 1$ ($C_t = c_t$ and $K_t = K$ in the warm-up).

Theorem 1 (Undetectability of the sampler)**.** *Consider a keyed construction in which, at every step t, the alphabet size K_t and the partition $B_{t,\cdot}$ are determined by the public prefix $V_{<t}$ and public parameters, every step with $K_t > 1$ embeds a symbol $C_t \in \mathbb{Z}_{K_t}$ masked with $\mathsf{RangePRF}_{k_m}$ at a public input that never repeats within a session, and the token is drawn by the sampler of Sec. 4.1, with $Y_t \sim p_t$ when $K_t = 1$. If $\mathsf{RangePRF}$ is computationally secure, the construction is computationally undetectable. Concretely, for every PPT distinguisher $\mathscr{A}$ in Definition 2 there is a PPT distinguisher $\mathscr{A}_1$ against $\mathsf{RangePRF}$ such that $\mathrm{Adv}^{\mathsf{steg}}_{\mathscr{A}}(\kappa) \leq \mathrm{Adv}^{\mathsf{prf}}_{\mathsf{RangePRF},\mathscr{A}_1}(\kappa)$.*

Theorem 1 states the undetectability of the warm-up protocol. Below, we analyze the error channel and soundness.

Corollary 1 (Substitution channel)**.** *In a construction as in Theorem 1, the receiver observes exactly one symbol $\widehat{C}_t$ per step with $K_t > 1$, in the same position as C_t, and $\widehat{C}_t \neq C_t$ if and only if the proposal at step t is rejected. Conditioned on $\mathscr{F}_{t-1}$ and the masked symbol A_t, $E_t \sim \mathrm{Bernoulli}\big((1 - K_t\pi_t(A_t))_+\big)$; if the mask is uniform and independent of $\mathscr{F}_{t-1}$, then $\mathbb{P}[E_t = 1 \mid \mathscr{F}_{t-1}] = \Delta_{K_t}(p_t)$.*

Theorem 2 (Soundness)**.** *If Tag is computationally secure, the scheme is sound (Definition 3) for every decoder. Concretely, if the receiver validates at most Q candidate messages ($Q = 1$ in the warm-up), there is a PPT distinguisher $\mathscr{A}_2$ against Tag such that $\mathsf{Extract}$ returns a message other than m or $\perp$ with probability at most $Q2^{-\lambda} + \mathrm{Adv}^{\mathsf{prf}}_{\mathsf{Tag},\mathscr{A}_2}(\kappa)$.*

4.3 Limitations of the Warm-Up

The warm-up is provably undetectable but leaves much of the capacity of the cover text unused. By Corollary 1, a symbol is wrong exactly when the proposal is rejected, which happens with probability $\Delta_K(p_t)$, the imbalance of the buckets under p_t. In the warm-up, however, the partition and the alphabet are fixed in advance without regard to p_t, and this costs capacity in two ways.

First, the partition is a random hash that ignores the probabilities of the tokens, so the heavy tokens of p_t may share a bucket, and the buckets are imbalanced even when p_t has enough entropy to fill them. For $K = 2$, if p_t puts mass 0.45 on each of two tokens, a random hash places both in the same bucket with probability $1/2$, which leaves $\Delta_K(p_t) \approx 0.45$ and the symbol wrong almost half of the time, whereas a partition that separates the two tokens is nearly balanced. Every such substitution costs redundancy in the ECC, and a partition balanced under p_t would avoid it, but the receiver cannot reproduce such a partition.

Second, the symbol alphabet is fixed, so every token carries $\log_2 K$ bits although its entropy varies from token to token. A high-entropy token could carry up to $\mathrm{H}(p_t)$ bits but is capped at $\log_2 K$, and a low-entropy token cannot carry even $\log_2 K$ bits, since no partition can balance a token that holds most of the mass, so its symbol only adds errors to the codeword. To maximize the channel capacity utilization, the alphabet should follow the entropy of p_t token by token.

Next section addresses the limitations with a small public draft model that gives both parties an estimate of p_t, from which they balance the buckets and choose the alphabet.

5 CODETTA-KEYED: Keyed Protocol

CODETTA-KEYED addresses both limitations of the warm-up with a small public draft model. The sender and the receiver run it on the public transcript and obtain $q_t = P_{\mathsf{Draft}}(\cdot \mid G_t(V_{<t}))$ as in Equation (2), so the sender and the receiver hold the same proxy for p_t. From q_t they balance the buckets and choose the alphabet size of every token (Sec. 5.1), and Sec. 5.2 presents the full protocol, whose alphabet-adaptive ECC produces the symbols on the fly in the alphabet chosen for each token. The sampler is inherited from the warm-up, so the undetectability carries over, and Sec. 5.3 shows how the draft model lowers the error rate of the channel. As in the warm-up, the two parties share a key $sk = (k_m, k_v)$, which Sec. 6 removes.

5.1 Draft-Balanced Buckets

At step t, both parties compute q_t and derive from it two public objects, the alphabet size K_t and a partition of $\mathscr{V}$ into K_t buckets (Algorithm 1, lines 4–5). Both are deterministic functions of q_t and public parameters, so the sender and the receiver obtain the same objects without communicating.

Bucket balancing. The first limitation of Sec. 4.3 is the imbalance of a random partition. To reduce $\Delta_K(p_t)$, we design a function $\mathsf{Bucket}(q_t, K)$ that partitions $\mathscr{V}$ into K buckets of nearly equal mass under q_t, and hence under p_t when the two distributions are close. It sorts the tokens by q_t in decreasing order and assigns each in turn to the bucket that is currently lightest, with ties broken by a public rule. We keep the notation of Sec. 4.1 for the map $f_{\mathsf{Bucket},t} : \mathscr{V} \to \mathbb{Z}_{K_t}$, its buckets $B_{t,a}$, and the masses $\pi_t(a) = p_t(B_{t,a})$ under the base model, and write $\mu_t(a) = q_t(B_{t,a})$ for the masses under the draft model and $\Delta_K(q_t) = \frac{1}{2}\sum_a |\mu_t(a) - \frac{1}{K}|$ for the imbalance under q_t, as in Equation (4).

Alphabet selection. To address the second limitation in Sec. 4.3, we design $\mathsf{ChooseAlphabet}(q_t)$, which returns the alphabet size K_t of step t. $1 < K_t \le K_{\max}$ is admissible only if q_t has enough entropy to fill K_t buckets and the partition $\mathsf{Bucket}(q_t, K_t)$ is close to balanced under q_t:

$$\mathrm{H}(q_t) \ge \log_2 K_t + \gamma_{\mathrm{H}}, \qquad \Delta_{K_t}(q_t) \le \theta, \tag{7}$$

where H is the Shannon entropy and $\gamma_{\mathrm{H}}, \theta$ are public thresholds. The entropy condition says that a symbol of $\log_2 K_t$ bits should not exceed the entropy of the token under the draft model, but entropy alone does not guarantee balance. A few heavy tokens can hold most of the mass of q_t while a long tail keeps $\mathrm{H}(q_t)$ large, and such tokens cannot be split across buckets, so even the best partition is imbalanced. The second condition checks the imbalance of the partition under the draft model. Among the admissible sizes, ChooseAlphabet returns the one with the highest score

$$\log_2 K_t - h(\widehat{e}_{K_t}) - \widehat{e}_{K_t} \log_2(K_t - 1), \tag{8}$$

where h is the binary entropy function. The score is an information proxy for the capacity of a K_t-ary symmetric channel with error rate $\widehat{e}_{K_t}$, a pre-defined parameter calibrated on held-out data for each alphabet size. If no alphabet size is admissible, then $K_t = 1$ is selected, which means that token at position t carries no symbol, and the sender samples $Y_t \sim p_t$ directly (Algorithm 1, line 7). The thresholds, $K_{\max}$, and the calibrated error rates $\widehat{e}_K$ are public.

Remarks. The symbol is embedded by the maximal coupling sampler of the warm-up (Algorithm 1, lines 13–15), so by Lemma 1 a symbol is substituted exactly when the proposal is rejected, which happens with probability $\Delta_{K_t}(p_t)$. The bucket masses of two distributions differ by at most their total variation, so by the triangle inequality

$$\Delta_{K_t}(p_t) \le \mathrm{TV}(p_t, q_t) + \Delta_{K_t}(q_t). \tag{9}$$

The first term is the mismatch between the base model and the draft model, which the ECC must absorb, and the second term is kept below θ by Equation (7). At a low-entropy step, a random hash leaves an imbalance near $1 - 1/K$, whereas ChooseAlphabet gives $K_t = 1$, so it never enters the ECC.

5.2 Alphabet-Adaptive ECC and the Full Protocol

Selecting K_t token by token raises a new coding problem. Both parties compute K_t from public information at every step, and we call a step with $K_t > 1$ usable; the sampler expects one symbol in $\mathbb{Z}_{K_t}$ at every usable step. A conventional ECC encodes the whole message into a codeword before transmission, but K_t is known only when generation reaches step t, so the codeword would have to be fixed before the alphabets of its symbols are known. Running generation ahead to learn the sizes does not help, because the tokens before step t determining K_t are the very tokens that carry the symbols, and a published token cannot be re-sampled without changing the transcript distribution. Restricting the alphabets to powers of two, so that a binary ECC can pack a variable number of bits into each token, loses the capacity of every step whose optimal alphabet sizes are not powers of two, which is rounded down.

We therefore design a novel alphabet-adaptive ECC that does not fix the codeword in advance. At each usable step it computes one symbol in $\mathbb{Z}_{K_t}$, and every error remains a substitution of one symbol and the number of symbols grows with the number of usable tokens. Algorithm 1 presents the full protocol, which combines the draft-balanced buckets and the alphabet selection of Sec. 5.1 with the alphabet-adaptive ECC. We first explain the idea behind the modular checks and then walk through Algorithm 1 in the order of the warm-up, encoding, masking and embedding, and decoding.

Modular checks. The heart of the alphabet-adaptive ECC is the modular check, which produces one check on the fly in the corresponding alphabet $\mathbb{Z}_{K_t}$ the step selects. Instead of transmitting the message itself, the sender answers one public question about it per usable token, and we call each question a check. The check of step t is defined only when generation reaches that step and K_t is known, and it maps the message to an element of $\mathbb{Z}_{K_t}$. For example, a token with $K_t = 3$ may carry the sum of two message bits modulo 3. Figure 2 illustrates the idea with three tokens of alphabet sizes 2, 3, and 5. The receiver collects the check values, a few of which are wrong because of rejections in the maximal coupling, and once it has enough of them, it solves for the message as one solves a system of equations with a few corrupted entries.

Algorithm 1 CODETTA-KEYED with draft-balanced buckets and an alphabet-adaptive ECC

Require: secret key $sk = (k_m, k_v)$; session identifier sid; public number of steps T, view G, checkpoints, and profile(s, K), the support and coefficients of check s over $\mathbb{Z}_K$

1: **procedure** EMBED(sk, sid, m)
2: $u \leftarrow m \,\|\, \mathsf{Tag}_{k_v}(\mathsf{sid}, m) \in \{0,1\}^{n+\lambda}$; $s \leftarrow 0$
3: **for** $t = 1, \ldots, T$ **do**
4: $q_t \leftarrow P_{\mathsf{Draft}}(\cdot \mid G_t(V_{<t}))$ ▷ public
5: $K_t \leftarrow \mathsf{ChooseAlphabet}(q_t)$; $B_{t,\cdot} \leftarrow \mathsf{Bucket}(q_t, K_t)$
6: $p_t \leftarrow P_{\mathsf{Base}}(\cdot \mid H_t, V_{<t})$ ▷ sender only
7: **if** $K_t = 1$ **then** sample $Y_t \sim p_t$ ▷ no symbol
8: **else**
9: $s \leftarrow s + 1$; $(S_s, a_{s,\cdot}) \leftarrow \mathrm{profile}(s, K_t)$
10: $C_s \leftarrow \sum_{i \in S_s} a_{s,i} u_i \bmod K_t$ ▷ check (10)
11: $A_s \leftarrow C_s + \mathsf{RangePRF}_{k_m}(\mathsf{sid}, s, K_t) \bmod K_t$
12: $\pi_t(a) \leftarrow p_t(B_{t,a})$ for all a ▷ bucket masses
13: sample $X_t \sim p_t(\cdot \mid B_{t,A_s})$ if $\pi_t(A_s) > 0$
14: $Y_t \leftarrow X_t$ w.p. $\min\{1, K_t \pi_t(A_s)\}$ ▷ accept
15: else $Y_t \sim \rho_t$ ▷ residual (6)
16: publish Y_t; the cover process updates H_{t+1}
17: **procedure** EXTRACT$(sk, \mathsf{sid}, V_{1:T})$
18: $s \leftarrow 0$; factor graph $\mathscr{G} \leftarrow \emptyset$; $\widehat{m} \leftarrow \perp$
19: **for** $t = 1, \ldots, T$ **do**
20: recompute q_t, K_t, $B_{t,\cdot}$ as in EMBED ▷ public
21: **if** $K_t > 1$ **then**
22: $s \leftarrow s + 1$; $r_s \leftarrow \mathsf{RangePRF}_{k_m}(\mathsf{sid}, s, K_t)$
23: $\widehat{C}_s \leftarrow f_{\mathsf{Bucket},t}(Y_t) - r_s \bmod K_t$ ▷ unmask
24: append the check $(S_s, a_{s,\cdot}, K_t, \widehat{C}_s)$ to $\mathscr{G}$
25: **if** t is a checkpoint **then**
26: $\widehat{m}' \leftarrow \mathrm{BP}(\mathscr{G})$; $\widehat{m} \leftarrow \widehat{m}'$ if its tag validates
27: **return** $\widehat{m}$

A check exists over any alphabet size, so the alphabet may change from token to token, and checks can be produced for as long as generation continues, so the code is rateless. A wrong token corrupts only its own check, and both parties derive K_t from the same public transcript and draft, so they agree on the usable tokens and on the index of every check. The channel therefore stays the substitution channel of Corollary 1, without insertions or deletions. Fountain codes [Lub02, Sho06] follow the same principle over a fixed alphabet, each symbol a random combination of message bits, and the modular checks extend it to alphabets that vary from symbol to symbol.

Encoding. We now walk through Algorithm 1. As in the warm-up, the sender first protects the message with a validation tag, $u = m \,\|\, \mathsf{Tag}_{k_v}(\mathsf{sid}, m) \in \{0,1\}^{n+\lambda}$, where n is the length of m and λ the length of the tag (line 2). The bits $u_1, \ldots, u_{n+\lambda}$ are the unknowns that the receiver aims to decode. Every usable step t produces one check, indexed by s (line 9), and each check is modular, a weighted sum of a few message bits reduced modulo K_t. A public ECC profile determines for check indexed s a degree d_s, a support $S_s \subseteq \{1, \ldots, n+\lambda\}$ of size d_s, and coefficients $a_{s,i} \in \{1, \ldots, K_t - 1\}$ for $i \in S_s$, and the sender computes the symbol (line 10)

$$C_s = \sum_{i \in S_s} a_{s,i} u_i \bmod K_t. \tag{10}$$

The profile draws the degrees from a public distribution that includes checks of degree one and two, because

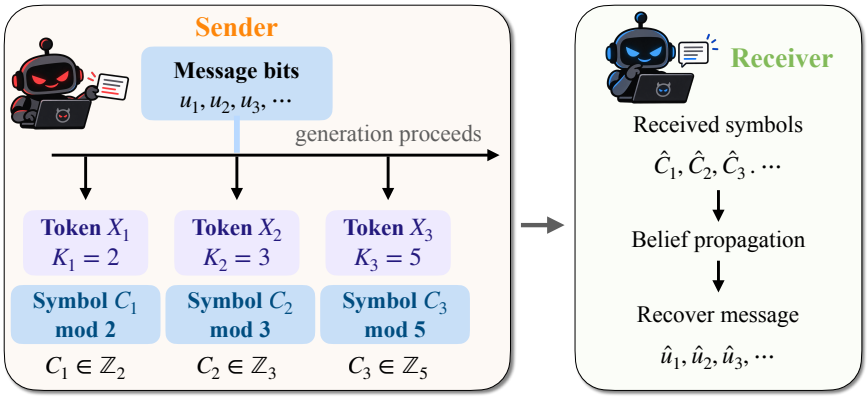

Figure 2: The alphabet-adaptive ECC. Every usable token carries one check of the message bits in the alphabet $\mathbb{Z}_{K_t}$ that the step selects, and the receiver decodes the message from the collected check values by belief propagation.

such checks pin down single bits and let decoding start reliably. The degree distribution and the support and coefficient rules are fixed before the communication starts, and they play a similar role to the parity-check matrix in LDPC.

Masking and embedding. The check value is not uniform, so the sender adds a fresh mask to it, as in the warm-up:

$$A_s = C_s + \mathsf{RangePRF}_{k_m}(\mathsf{sid}, s, K_t) \bmod K_t, \tag{11}$$

where the mask is a pseudorandom element of $\mathbb{Z}_{K_t}$. Since the alphabet differs from check to check, RangePRF takes K_t as an input. The masked symbol A_s is the target bucket of step t. The sender proposes $X_t \sim p_t(\cdot \mid B_{t,A_s})$, accepts it with probability $\min\{1, K_t \pi_t(A_s)\}$, and otherwise samples the residual ρ_t of Equation (6) (lines 13–15), exactly as in the warm-up. Compared with the warm-up, only two inputs have changed. The buckets are now determined by the draft distribution, and the symbol comes from the modular check.

Decoding. The receiver first replays the public computation (line 20). It runs the draft model on the transcript, recovers q_t, K_t, and the buckets at every step, and reads the masked check value $\widehat{A}_s = f_{\mathsf{Bucket},t}(Y_t)$ at every usable token (line 23). It then subtracts the mask of Equation (11) to obtain $\widehat{C}_s$ (line 23). Let t_s be the step of check s and M the number of checks in the transcript. The receiver now holds a system of M equations in the $n + \lambda$ unknown bits,

$$\sum_{i \in S_s} a_{s,i} u_i \equiv \widehat{C}_s \pmod{K_{t_s}}, \quad s = 1, \ldots, M, \tag{12}$$

in which every equation whose proposal was rejected by the maximal coupling has a wrong $\widehat{C}_s$ (line 24). Decoding therefore reduces to solving a sparse system of modular equations with a few corrupted entries, and belief propagation is a conventional tool for this problem, as it is for LDPC codes (Sec. 2). The receiver builds a factor graph with one variable node per bit and one factor per equation. The factor of equation s assigns to each local assignment u_{S_s} the likelihood of the observed value $\widehat{C}_s$ under the error rate $\widehat{e}_{K_{t_s}}$ calibrated in Sec. 5.1, namely $1 - \widehat{e}_{K_{t_s}}$ if u_{S_s} satisfies equation s of (12) and $\widehat{e}_{K_{t_s}}/(K_{t_s} - 1)$ otherwise. See Appendix A for the details of belief propagation (BP). A checkpoint is a public step at which the receiver attempts a decode. At each checkpoint, the receiver runs BP on the equations collected so far and obtains a candidate message $\widehat{m}$ (line 26). It accepts the candidate only if the decoded tag equals $\mathsf{Tag}_{k_v}(\mathsf{sid}, \widehat{m})$. The sender can run EXTRACT on the public transcript itself, so it knows when the decode validates. It keeps embedding checks until then, across responses if one response is too short.

5.3 Undetectability and Soundness

CODETTA-KEYED has the form analyzed in Sec. 4.2. The alphabet size K_t and the partition $B_{t,\cdot}$ are deterministic functions of the public prefix (Algorithm 1, lines 4–5); the symbol of a usable step is the check C_s, masked with $\mathsf{RangePRF}_{k_m}(\mathsf{sid}, s, K_t)$, where the check index s is determined by the prefix and never repeats within a session. The maximal coupling sampler is that of the warm-up (lines 13–15), and a step with $K_t = 1$ samples $Y_t \sim p_t$ (line 7). Theorems 1 and 2 and Corollary 1 therefore apply as stated.

Corollary 2 (Undetectability of CODETTA-KEYED)**.** *If* RangePRF *is computationally secure,* CODETTA-KEYED *is computationally undetectable, with the bound of Theorem 1.*

Recall that check indexed s is carried by token at step t_s, and let $E_s = \mathbf{1}\{\widehat{C}_s \neq C_s\}$ indicate a substituted check.

Corollary 3 (Error rate of CODETTA-KEYED)**.** *Under uniform masks, check s is substituted exactly when the proposal at step t_s is rejected, and by Corollary 1 and Equation (9),*

$$\mathbb{P}[E_s = 1 \mid \mathscr{F}_{t_s-1}] = \Delta_{K_{t_s}}(p_{t_s}) \leq \mathrm{TV}(p_{t_s}, q_{t_s}) + \theta,$$

where θ is the balance threshold that ChooseAlphabet *enforces on the public partition in Equation (7).*

Corollary 4 (Soundness of CODETTA-KEYED)**.** *Theorem 2 holds for* CODETTA-KEYED *with Q the number of checkpoints, as each checkpoint validates one candidate.*

Remarks. Since the sender keeps embedding checks until the decode validates (Sec. 5.2), a wrong message is returned only with the soundness probability of Corollary 4, and the substitution rate of Corollary 3 determines the number of tokens that a message costs and Sec. 7 measures this cost. Compared with the warm-up, substitutions are charged at usable tokens only, and their rate is at most $\mathrm{TV}(p_t, q_t) + \theta$ instead of the imbalance of a random partition, and at most θ when the draft matches the base model.

6 CODETTA-KE: Key Exchange

The protocols so far assume that the sender and the receiver share a key. Independently deployed agents, however, do not have such a key, and they cannot establish one in the open, because a public key or a ciphertext in a conversation is an obvious signal to the auditor. The key exchange itself must therefore be hidden in ordinary text without a key.

Without a key, the sender cannot mask the symbols as in the keyed protocols, so the symbols themselves must be uniform. To embed a symbol, the sender draws candidate blocks of tokens from its LLM distribution and publishes a block whose public hash equals the symbol, and the receiver recomputes the hash to read it. Since the hash values of a finite pool are never perfectly balanced, a sender that keeps the transcript distribution exact publishes a wrong symbol with a constant probability. The construction of [VZ26] follows this route and tolerates the noise with a feedback compiler that signals the exchange bit by bit. CODETTA-KE keeps the same exact sampling of blocks but transmits lattice coefficients instead of bits to improve efficiency. The messages of an MLWE key exchange are pseudorandom vectors over $\mathbb{Z}_q$, so each carrier position transmits one coefficient as a q-ary symbol of about $\log_2 q$ bits. We design an exact-cover carrier so that a wrong symbol displaces the coefficient by a small amount, and MLWE absorbs this displacement as extra noise, with a short ECC corrects the residual errors.

We present the carrier in Sec. 6.1 and the key exchange in Sec. 6.2, analyze undetectability and correctness in Sec. 6.3, and give the concrete parameters in Sec. 6.4.

Algorithm 2 Lifted exact-cover carrier at one position

Require: modulus q; session identifier sid; replica ratio β and pool size $L = \beta q$; hash Hash_q with range $\mathbb{Z}_q$; cyclic distance $|u - v|_q$ modulo q; all public

1: **procedure** PUBLISH(i, a) ▷ sender; symbol $a \in \mathbb{Z}_q$
2: $p_i \leftarrow$ next-block distribution ▷ sender only
3: sample $X_0, \ldots, X_{L-1} \sim p_i$ independently ▷ pool
4: $y_j \leftarrow \mathsf{Hash}_q(\mathsf{sid}, i, X_j)$ for $j < L$ ▷ public labels
5: $\pi \leftarrow$ permutation with $y_{\pi(0)} \le \cdots \le y_{\pi(L-1)}$
6: $r^\star \leftarrow \arg\min_{r<L} \sum_{k<L} \left| \lfloor k/\beta \rfloor - y_{\pi(k+r \bmod L)} \right|_q^2$
7: $\mathsf{Assign}(a', J) \leftarrow \pi(a'\beta + J + r^\star \bmod L)$
8: sample $J \sim \mathscr{U}(\{0, \ldots, \beta - 1\})$ ▷ replica
9: publish $X_{\mathsf{Assign}(a,J)}$ ▷ uniform over the pool
10: **procedure** READ(i, X)
11: **return** $\mathsf{Hash}_q(\mathsf{sid}, i, X)$ ▷ label of X

6.1 Lifted Exact-Cover Carrier

As described above, the carrier embeds each symbol with a public hash and samples the published block to ensure that the transcript preserves the original distribution. A carrier position is a block of at most B tokens, and this block transmits one coefficient $a \in \mathbb{Z}_q$ of a lattice message. We call the agent that publishes the block the sender and Algorithm 2 gives the procedure at one position.

Pool and labels. At position i, the sender draws $L = \beta q$ candidate blocks $X_0, \ldots, X_{L-1}$ independently from p_i, its honest distribution over the next block, where $\beta \ge 1$ is a public replica ratio (line 3). The pool does not depend on the symbol a, and the receiver never sees the pool. Every block gets a label from a public hash function $y_j = \mathsf{Hash}_q(\mathsf{sid}, i, X_j) \in \mathbb{Z}_q$ (line 4). The hash takes the session identifier and the position as inputs, so labels at different positions and in different exchanges are independent. The receiver can compute the labels of received blocks (line 11).

Selection. Directly publishing a block with label a would bias the transcript, because the labels of a finite pool are not balanced. The sender instead fixes a bijection Assign between the L slots (a, J), with $J \in \{0, \ldots, \beta - 1\}$, and the L blocks. It draws J uniformly and publishes $X_{\mathsf{Assign}(a,J)}$ (line 9). When a is uniform, (a, J) is uniform over the slots, and the bijection makes the published index uniform over the pool. The published block is therefore an honest sample from p_i, whatever the labels are (Lemma 2). Every block belongs to exactly one slot, which is the exact cover, and the replica index lifts the q labels to L slots. This selection follows the bundle sampler of [VZ26], extended from one bit to a q-ary symbol.

Choosing the bijection. Undetectability holds for every bijection, so the choice of Assign only affects the carrier error, and we choose it to keep the error small. The sender sorts the blocks by label (line 5) and lists the slots by symbol, so that slot (a, J) has rank $a\beta + J$. It then aligns the two lists up to a cyclic rotation and keeps the rotation with the least total squared distance between slot symbols and block labels (lines 6–7). A cyclic correlation finds the best rotation in $O(L \log L)$ time. When the labels are nearly balanced, almost every slot gets a block with its own label, and the other slots get a block whose label is close. A carrier error is thus a small displacement of a modulo q.

Serializer. Some phases of the exchange need less precision. For them, a public serializer $g : \mathbb{Z}_q \to \mathscr{Y}$ maps each symbol to a label in a smaller alphabet $\mathscr{Y}$, and a public map $g^+ : \mathscr{Y} \to \mathbb{Z}_q$ lifts a label back to a symbol. Algorithm 2 then hashes the blocks into $\mathscr{Y}$, compares the label of a block with $g(\lfloor k/\beta \rfloor)$ instead of $\lfloor k/\beta \rfloor$ in line 6, and READ returns $g^+(y)$ for the label y it reads. The slots are still indexed by a, so the published block distribution is the same as p_i.

6.2 Key Exchange

CODETTA-KE runs an MLWE key encapsulation (Sec. 2) in three phases, in which the receiver publishes a pseudorandom share and the sender then publishes a vector and the masked pre-key. Every message is sent coefficient by coefficient through the carrier of Sec. 6.1, and Algorithm 3 lists the steps. We name the parties by their roles in CODETTA-KEYED.

Algorithm 3 CODETTA-KE: three-phase key exchange over the lifted exact-cover carrier

Public parameters. Session identifier sid, matrix $A \in R_q^{k\times k}$, distributions χ_{sec} and χ_{err}, ECC $(\mathsf{ECC}, \mathsf{DecECC})$ with codeword length N, serializers $g_b = \mathsf{id}$, g_w, g_z, and KDF. Every arrow sends one coefficient per carrier position through Algorithm 2, and V is the public transcript of the three phases.

	Sender		**Receiver**
	Setup		
1	$s_{\text{S}} \sim \chi_{\text{sec}}^k$, $e_1 \sim \chi_{\text{err}}^k$, $e_2 \sim \chi_{\text{err}}$		$s_{\text{R}} \sim \chi_{\text{sec}}^k$, $e_{\text{R}} \sim \chi_{\text{sec}}^k$
	Phase 1, the receiver's share		
2			$b \leftarrow As_{\text{R}} + e_{\text{R}} \in R_q^k$
	read $\widetilde{b} \in R_q^k$	$\longleftarrow$ b (kd positions, g_b) $\longleftarrow$	
	Phases 2 and 3, the sender's response		
3	$K_0 \sim \mathscr{U}(\{0,1\}^\ell)$; $c^+ \leftarrow \mathsf{ECC}(K_0) \,\|\, 0^{d-N}$		
4	$w \leftarrow A^T s_{\text{S}} + e_1 \in R_q^k$		
		w (kd positions, g_w) $\longrightarrow$	read $\widetilde{w} \in R_q^k$
5	$z \leftarrow \widetilde{b}^T s_{\text{S}} + e_2 + \iota(c^+) \in R_q$		
		z (d positions, g_z) $\longrightarrow$	read $\widetilde{z} \in R_q$
6			$\omega \leftarrow \widetilde{z} - s_{\text{R}}^T \widetilde{w}$
7			$\widehat{c} \leftarrow \mathsf{HD}(\omega)_{[N]}$; $\widehat{K}_0 \leftarrow \mathsf{DecECC}(\widehat{c})$
	Key derivation and confirmation		
8	$K_{\text{S}} \leftarrow \mathsf{KDF}(K_0, V)$		$K_{\text{R}} \leftarrow \mathsf{KDF}(\widehat{K}_0, V)$
9	embed a fixed message under K_{S}		
		CODETTA-KEYED $\longrightarrow$	validate under K_{R}, else restart

Setup. Let $R_q = \mathbb{Z}_q[x]/(x^d+1)$ be the polynomial ring of degree d, let k be the module rank, so that an element of R_q^k has kd coefficients and occupies kd carrier positions, and let $A \in R_q^{k\times k}$ be a public random matrix. The session identifier sid is fresh for every exchange and enters every hash of the carrier. The receiver samples a secret s_{R} and an error e_{R}, and the sender samples a secret s_{S} and errors e_1 and e_2, from the distributions χ_{sec} and χ_{err} over polynomials with small coefficients (step 1 of Algorithm 3; see Sec. 6.4). The exchange sends three messages through the carrier, the receiver's share $b \in R_q^k$, the sender's vector $w \in R_q^k$, and the masked pre-key $z \in R_q$, which occupy kd, kd, and d positions. A party reads each message with a small displacement; we write the values read as $\widetilde{b}$, $\widetilde{w}$, and $\widetilde{z}$, and the displacements as $\zeta_b = \widetilde{b} - b$, $\zeta_w = \widetilde{w} - w$, and $\zeta_z = \widetilde{z} - z$.

The receiver's share. The receiver computes $b = As_{\text{R}} + e_{\text{R}}$ and publishes its kd coefficients (step 2). The share is an MLWE sample, hence pseudorandom, so the carrier can transmit its coefficients as uniform symbols. The sender reads $\widetilde{b} = b + \zeta_b$ through the identity serializer (step 2).

The sender's response. The sender samples a pre-key $K_0 \sim \mathscr{U}(\{0,1\}^\ell)$ and encodes it as $c = \mathsf{ECC}(K_0) \in \{0,1\}^N$ with $N \le d$, using an ECC that corrects up to τ flipped bits. It pads c with $d-N$ zero bits to $c^+ \in \{0,1\}^d$ and maps it to the ring element $\iota(c^+) \in R_q$, where ι places a 0 bit at coefficient 0 and a 1 bit at coefficient $\lfloor q/2 \rceil$ (step 3). It then computes $w = A^T s_{\text{S}} + e_1$, an MLWE sample under its secret, and $z = \widetilde{b}^T s_{\text{S}} + e_2 + \iota(c^+)$, which hides the encoded pre-key under a mask from its secret and the share it read (steps 4 and 5). It publishes the kd coefficients of w with serializer g_w and the d coefficients of z with g_z.

Lemma 3 proves that b, w, and z are jointly pseudorandom.

Receiver decoding. The receiver reads $\widetilde{w} = w + \zeta_w$ and $\widetilde{z} = z + \zeta_z$ through g_w^+ and g_z^+ and computes $\omega = \widetilde{z} - s_{\mathrm{R}}^T \widetilde{w} \pmod q$ (step 6). By the definitions of b, w, and z, the term $s_{\mathrm{R}}^T A^T s_{\mathrm{S}}$ appears in both $\widetilde{z}$ and $s_{\mathrm{R}}^T \widetilde{w}$ and cancels, which leaves $\iota(c^+)$ plus a noise term (Theorem 4). Every noise term is a small error or displacement, or one times a small secret, so the noise is far below $q/2$ in almost every coefficient. The receiver applies the hard-decision map $\mathsf{HD} : R_q \to \{0,1\}^d$, which rounds each coefficient to the nearer of 0 and $\lfloor q/2 \rceil$, keeps the first N bits as $\widehat{c} = \mathsf{HD}(\omega)_{[N]}$, and decodes $\widehat{K}_0 = \mathsf{DecECC}(\widehat{c})$ (step 7). A coefficient is decided wrongly only when its noise reaches about $q/4$, and the ECC corrects up to τ of them.

Key derivation and confirmation. Both parties derive the session key from the pre-key and the public transcript V, $K_{\mathrm{S}} = \mathsf{KDF}(K_0, V)$ and $K_{\mathrm{R}} = \mathsf{KDF}(\widehat{K}_0, V)$, which agree whenever $\widehat{K}_0 = K_0$ (step 8). The parties then run CODETTA-KEYED under their keys to transmit a fixed public message (step 9). If the tag validates, the keys agree; otherwise they repeat the exchange with fresh randomness. Attempts fail independently, so r attempts all fail with probability at most p_{fail}^r, for example below 10^{-6} after three attempts when one attempt succeeds with probability 0.99.

6.3 Undetectability and Correctness

This section shows that the transcript of CODETTA-KE is indistinguishable from an honest transcript, that the derived key is pseudorandom, and that both parties obtain the same key whenever the carrier errors stay inside a decoding region. We model the carrier hash and the KDF as random oracles, which the passive auditor of Sec. 3.2 may query at any time. Since the two parties never see these queries, an auditor that queries during the exchange has the same view as one that queries after the transcript is complete, so we analyze the latter.

Notation. Let $\mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}}(\kappa)$ be the advantage of $\mathscr{B}$ in telling $(A, As_{\mathrm{R}} + e_{\mathrm{R}})$ from (A,U) with $A \sim \mathscr{U}(R_q^{k\times k})$ and $U \sim \mathscr{U}(R_q^k)$, and $\mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}}(\kappa)$ its advantage in telling $(M, Ms_{\mathrm{S}} + e)$ from (M,U') with $M \sim \mathscr{U}(R_q^{(k+1)\times k})$, $e = (e_1, e_2)$, and $U' \sim \mathscr{U}(R_q^{k+1})$. For a passive auditor $\mathscr{A}$, let $\mathrm{Adv}^{\mathsf{cover}}_{\mathscr{A}}(\kappa)$ be its advantage in distinguishing the public transcript from an honest one, and $\mathrm{Adv}^{\mathsf{key}}_{\mathscr{A}}(\kappa)$ its advantage in additionally distinguishing the sender's key from a uniform κ-bit string. For a distribution $\mathscr{D}$ over complete sessions, let $\delta_{\mathscr{D}} = \mathbb{P}_{\Omega\sim\mathscr{D}}[K_{\mathrm{R}} \neq K_{\mathrm{S}}]$ be the probability that the parties end with different keys.

Undetectability. Lemma 2 shows that the carrier outputs an honest transcript when the symbols it embeds are uniform, and Lemma 3 shows that the symbols CODETTA-KE embeds, the coefficients of b, w, and z, are computationally indistinguishable from uniform. Theorem 3 combines the two lemmas and give the security argument.

Lemma 2 (Labeled lifted exact cover)**.** *If the symbol a is uniform on $\mathbb{Z}_q$ and independent of the pool, the published block has distribution p_i. If the symbols of all positions are independent and uniform, and independent of the pools, the published blocks are distributed as an honest transcript.*

Lemma 3 (Joint pseudorandomness of the transmitted symbols)**.** *Let $\mathscr{A}$ be a PPT auditor that reads the completed transcript. Feeding the carrier independent uniform symbols in place of the coefficients of b, w, and z changes the probability that $\mathscr{A}$ outputs 1 by at most $\mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}(\kappa) + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}(\kappa)$, for PPT distinguishers $\mathscr{B}_1$ and $\mathscr{B}_2$ built from $\mathscr{A}$.*

Theorem 3 (Undetectability and key security)**.** *For every PPT passive auditor $\mathscr{A}$ with at most Q_{KDF} KDF queries, there are PPT $\mathscr{B}_1$ and $\mathscr{B}_2$ with*

$$\mathrm{Adv}^{\mathsf{cover}}_{\mathscr{A}}(\kappa) \le \mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}(\kappa) + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}(\kappa),$$

$$\mathrm{Adv}^{\mathsf{key}}_{\mathscr{A}}(\kappa) \le \mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}(\kappa) + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}(\kappa) + \frac{Q_{\mathsf{KDF}}}{2^{\ell}}.$$

Correctness. Correctness has two parts. Theorem 4 gives a condition on the carrier errors under which the receiver recovers the key, and Theorem 5 bounds how often an execution of a given workload violates this condition.

Theorem 4 (Noise-resilient decoding)**.** *For arbitrary displacements $(\zeta_b, \zeta_w, \zeta_z)$, where ζ_w and ζ_z include the serializer reconstruction error, the receiver's residual is*

$$\omega = \iota(c^+) + (e_R + \zeta_b)^T s_S - s_R^T(e_1 + \zeta_w) + e_2 + \zeta_z \pmod q,$$

and if $W = d_H(\mathsf{HD}(\omega)_{[N]}, c) \le \tau$, the decoder recovers K_0 and $K_R = K_S$.

Corollary 5 (Workload-specific correctness)**.** *For every distribution $\mathscr{D}$ over the agent outputs, $\delta_{\mathscr{D}} \le \mathbb{P}_{\Omega\sim\mathscr{D}}[W > \tau]$.*

Since the displacement depends on the language model and the task, we estimate $\mathbb{P}_{\mathscr{D}}[W > \tau]$ empirically.

Theorem 5 (Empirical correctness)**.** *Let $\Omega_1, \ldots, \Omega_m \sim \mathscr{D}$ be independent complete sessions and $S < m$ the number of them with $W > \tau$. With probability at least $1 - \alpha$ over the draw of the m sessions,*

$$\delta_{\mathscr{D}} \le \mathbb{P}_{\mathscr{D}}[W > \tau] \le \mathrm{Beta}^{-1}(1 - \alpha; S + 1, m - S).$$

6.4 Instantiation

The results of Sec. 6.3 hold for any choice of the lattice parameters, the serializers, and the ECC. This section fixes the choice used in the evaluation and states the security level it provides. The engineering parameters of the carrier, such as the replica ratios and the block length, do not change the proofs, and Sec. 7.1 gives them.

Lattice parameters. We use the parameters of ML-KEM-512 [Nat24]. Specifically, we use the ring R_q with $d = 256$ and $q = 3329$, rank $k = 2$, $\chi_{\mathrm{sec}} = \mathsf{CBD}_3$, and $\chi_{\mathrm{err}} = \mathsf{CBD}_2$, where CBD_η is the centered binomial distribution of parameter η. The two assumptions of Theorem 3, decisional MLWE with k samples for the share and with $k+1$ samples for the response, are the two assumptions in the IND-CPA security proof of ML-KEM-512, at the same ring, modulus, rank, and noise distributions [Nat24]. The exchange therefore relies on the same MLWE problems as ML-KEM-512, which NIST places in security category 1, at least as hard to break as exhaustive key search on AES-128.

Serializers. The share b is sent with the identity serializer, as Lemma 3 requires. The response uses $g_w = \mathrm{Compress}_{10}$ and $g_z = \mathrm{Compress}_4$, the ciphertext compression of ML-KEM-512, and the hard-decision map HD is $\mathrm{Compress}_1$.

Pre-key and ECC. The pre-key has $\ell = 128$ bits, which matches the security category and makes the term $Q_{\mathsf{KDF}}/2^\ell$ in Theorem 3 negligible at the same level, and the session key has $\kappa = 256$ bits. The ECC must place its codeword in the $d = 256$ coefficients of one ring element and carry at least ℓ information bits, and within these two constraints we take the largest correction radius. The binary BCH code of length 255 and dimension 131 (Appendix A) corrects $\tau = 18$ errors and meets both constraints. The exchange occupies $kd + kd + d = 1{,}280$ carrier positions.

7 Evaluation

We summarize the experimental setup in Sec. 7.1 and compare the performance of CODETTA-KEYED with ARS [BYP+25] and the warm-up in Sec. 7.2. We then perform ablations of CODETTA-KEYED in Sec. 7.3, test its robust variant against an auditor that edits the text in Sec. 7.4, and measure the correctness and the cost of CODETTA-KE in Sec. 7.5.

7.1 Experimental setup

Agent workloads. We evaluate on three simulated agent workloads and give their details in Appendix B. In each, the sender runs a large base model on a private context, and the receiver sees only the public transcript. FINANCIAL simulates the second scenario of Sec. 3.1, in which the receiver asks about a synthetic issuer and the sender answers from its private database records. IFEVAL [ZLM+23] is a benchmark of natural-language requests with constraints on length, structure, wording, and formatting, and the sender's private context is a plan it drafts before answering. CTF simulates the first scenario, in which the sender solves picoCTF challenges [CBB14, YPNY23] with shell and Python tools in a sandbox and then writes a technical solution report, the cover text, to a file system that the receiver reads. The sender's private context is the solve trajectory. In FINANCIAL and IFEVAL, the agents communicate in a direct conversation. A draft-assisted receiver runs a small public draft model on the public transcript, and a model-free receiver does not run a language model.

Models. The senders are the widely used open-source models Qwen2.5-7B [YYZ+24], Llama-3.1-8B [GDJ+24], and Ministral-3-8B [LKS+26], and the public drafts are Qwen2.5-1.5B, Llama-3.2-1B, and Ministral-3-3B, all in their Instruct variants. All experiments run on one NVIDIA H200 GPU.

Baselines. ARS [BYP+25] is the steganographic protocol that achieves the best performance under asymmetric resources, and we run its official implementation with the suggested parameters. Our second baseline is the warm-up of Sec. 4, with a fixed binary alphabet and a highly optimized rateless LDPC code from NVIDIA Sionna [HCAA+22], the 5G NR mother code of rate 0.2, whose symbols are sent incrementally until the receiver decodes.

Messages. CODETTA and the warm-up carry 1,024 payload bits per message and append a 32-bit tag. ARS delivers far lower capacity, so we give it a 32-bit payload on every workload, a choice that favors ARS. It embeds one bit at a time and cannot repair a wrong bit, so a longer payload takes proportionally more tokens and fails more often, which only lowers its capacity. Each reported number averages over 40 messages on FINANCIAL, 477 on IFEVAL, and 65 on CTF. A message may span several responses. The payloads of every method are drawn from one fixed seed, and when a workload runs out of requests we rotate through its request pool again.

Metrics. Our main metric is capacity, the number of decoded payload bits divided by the number of transmitted tokens, i.e., the end-to-end payload rate. We also report encoding and decoding throughput in tokens per second, with sender batch size 10. Encoding throughput includes generation, the draft forward pass, and embedding, and decoding throughput covers the replay of the draft model over the transcript and the ECC decoding.

Key exchange. Section 6.4 gives the instantiation details of CODETTA-KE. We run CODETTA-KE with the three sender models on the three workloads, with token blocks of at most $B = 64$ tokens, and bound its failure probability with Theorem 5 over $m = 100{,}000$ simulated sessions. A simulated session runs the lattice operations of CODETTA-KE with fresh randomness and draws the displacement of every position independently from the distribution that we measure on the sender model's candidate pools. We use the lattice parameters of ML-KEM-512, a 128-bit pre-key, and a binary BCH code of length 255 that corrects 18 errors, so one exchange occupies 1,280 carrier positions. The carrier uses the first complete response block of a turn as its token block, capped at $B = 64$ tokens, and the replica ratios are $(\beta_b, \beta_w, \beta_z) = (4, 4, 1)$. The masked pre-key z is sent through $\mathrm{Compress}_4$, which has only 16 labels per position, so one replica suffices there.

7.2 Performance of keyed protocols

We report the capacity of the three methods in Table 1, and all the messages are successfully decoded. CODETTA-KEYED significantly improves the capacity and delivers 4.0 to 6.3 times that of the warm-up and

	FINANCIAL			IFEVAL			CTF		
Method	Qwen	Llama	Ministral	Qwen	Llama	Ministral	Qwen	Llama	Ministral
Capacity (bits per token)									
CODETTA-KEYED	**0.1261**	**0.2411**	**0.3557**	**0.1734**	**0.2882**	**0.4486**	**0.0352**	**0.2597**	**0.2611**
Warm-up	0.0282	0.0381	0.0875	0.0361	0.0505	0.1117	0.0077	0.0504	0.0586
ARS	0.0015	0.0115	0.0111	0.0027	0.0088	0.0141	0.0003	0.0124	0.0075
Gain of CODETTA-KEYED									
over Warm-up	4.5×	6.3×	4.1×	4.8×	5.7×	4.0×	4.6×	5.2×	4.5×
over ARS	83.0×	21.0×	32.0×	64.6×	32.7×	31.8×	102.2×	20.9×	35.0×
Encoding throughput (tokens per second)									
CODETTA-KEYED	244	204	155	212	350	152	72	85	56
Warm-up	306	249	203	331	472	205	84	90	63
ARS	248	207	172	258	340	172	84	91	63
Decoding throughput (tokens per second)									
CODETTA-KEYED	1.0k	1.3k	0.7k	0.5k	1.0k	0.8k	0.2k	0.4k	0.2k
Warm-up	381k	393k	391k	251k	341k	363k	16k	14k	17k
ARS	7.5k	8.0k	8.0k	7.5k	8.0k	8.0k	42k	38k	41k

Table 1: Capacity, gain of CODETTA-KEYED over each baseline, and encoding and decoding throughput, for the three sender models on the three workloads (Sec. 7.1). Warm-up is the construction of Sec. 4. Every message is correctly decoded.

21 to 102 times that of ARS. We observe consistent gains over the baselines across workloads and models. Sec. 7.3 breaks down the improvement over the warm-up by component.

The absolute capacity follows the entropy of the cover text and can differ by an order of magnitude across workloads and models. Ministral writes the most varied text and reaches 0.36 to 0.45 bits per token on FINANCIAL and IFEVAL. Qwen on CTF is the other extreme. Its reports copy long spans of the private solve trajectory verbatim, 60% of the tokens in the report have an entropy below 0.1 bits, and every method drops below 0.04 bits per token there. Even there, CODETTA-KEYED delivers 4.6 times the capacity of the warm-up and 102 times the capacity of ARS.

Throughput. The throughput in Table 1 shows that the sender pays little for the capacity gain of CODETTA-KEYED. The CODETTA-KEYED sender runs the draft model once for every token, on top of the base model. It still runs at 82% to 103% of the speed of the ARS sender, and at 85% or above in all but one setting, because the base model takes most of the generation time and the draft is much smaller. The warm-up sender is the fastest, since it samples from the base model without a draft.

The receiver pays more, since it replays the draft model over the whole transcript, while the two baselines decode with hashes alone and reach thousands to hundreds of thousands of tokens per second. The CODETTA-KEYED receiver still decodes at 240 to 1,260 tokens per second, 2 to 6 times faster than the sender generates. Considering the significant capacity improvement, we believe the decoding speed cost is acceptable, and the receiver keeps pace with the sender.

7.3 Ablations of CODETTA-KEYED

In this section, we measure which components of Sec. 5 account for the gain over the warm-up on the FINANCIAL workload. We also ask whether a receiver that sees a larger portion of the private context or runs a stronger draft delivers a higher capacity, and whether the draft must come from the sender's model family.

Components. In Figure 3, we build CODETTA-KEYED up from the warm-up one component at a time, and

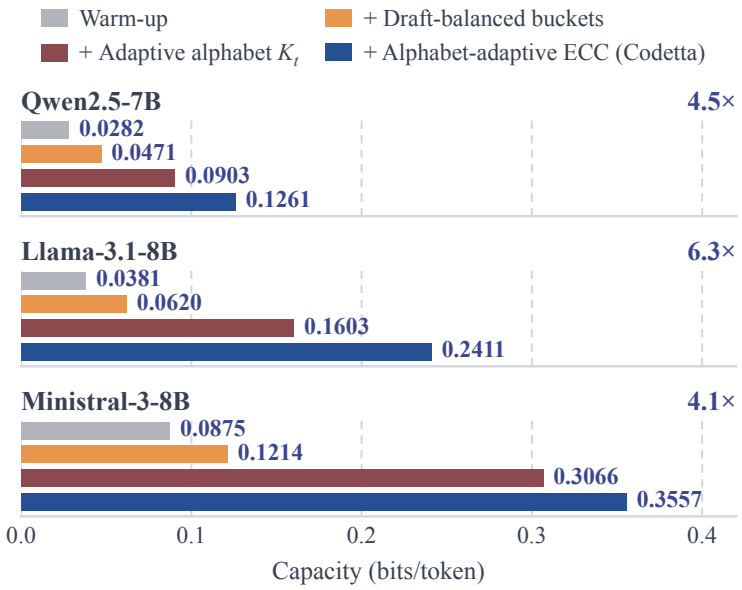

Figure 3: Component ablation on FINANCIAL. The top bar is the warm-up of Sec. 4, each bar below adds one component, and the bottom bar is CODETTA-KEYED. The top right number is the gain of CODETTA-KEYED over warm-up.

every component raises the capacity. Draft-balanced buckets (Sec. 5.1), with the alphabet fixed at $\mathbb{Z}_2$, raise the capacity by 39% to 67%. Letting the alphabet follow the entropy of each token raises it by another 92% to 159%, even though the LDPC code still limits the alphabet to powers of two. The alphabet-adaptive ECC (Sec. 5.2) lifts this limit and adds a further 16% to 50%, the capacity that a binary code leaves unused when it rounds the alphabet down. Together, the three components deliver 4.1 to 6.3 times the capacity of the warm-up.

Shared context. In Figure 4, we give the receiver a growing portion of the sender's private records in FINANCIAL, in which the receiver sees none of them, to full sharing of the private context, 1,597 tokens on average. The sender is Qwen2.5-7B, and we compare the 1.5B draft of the main experiments with a 7B draft that is the same as the sender LLM. Capacity grows with the shared context for both drafts, from 0.126 to 0.152 bits per token with the 1.5B draft and from 0.129 to 0.206 with the 7B draft.

The larger draft helps more as more context is shared. Without shared records, the two drafts are on par, at 0.126 and 0.129 bits per token, since the 7B draft still has to estimate the distribution without seeing the private records. At half sharing, the 7B draft delivers 13% more than the 1.5B draft, and at full sharing 36% more, where it has both the sender's model and the sender's complete context, while the 1.5B draft still differs in model. Both trends come from a draft that tracks the sender's distribution more closely, so that fewer proposals are rejected.

Model mismatch. In Table 2, we pair each sender with drafts from different families, and cross-family pairs use a public byte-level bridge between the two tokenizers. The capacity varies more with the sender than with the draft. Ministral keeps 0.30 to 0.36 bits per token with every draft, and Qwen stays between 0.11 and 0.13. All the capacity results are close for the same sender base model with a same-family draft being the best for Qwen and Ministral, and a cross-family draft losing 4% to 16%. For Llama, the Ministral-3-3B draft is on par with the same-family draft, and the Qwen2.5-1.5B draft loses 5%.

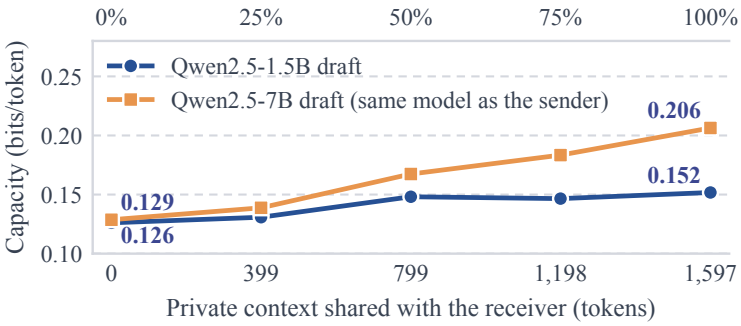

Figure 4: Capacity on FINANCIAL with Qwen2.5-7B sender as the receiver sees a growing fraction of private context, for different draft models. The upper axis is the shared fraction.

	Draft		
Sender	Qwen2.5-1.5B	Llama-3.2-1B	Ministral-3-3B
Qwen2.5-7B	0.1261	0.1112	0.1217
Llama-3.1-8B	0.2299	0.2411	0.2428
Ministral-3-8B	0.3308	0.3002	0.3557

Table 2: Capacity of CODETTA-KEYED on FINANCIAL for various sender–draft pairs. Cross-family pairs use a public tokenizer bridge. Every message is correctly decoded.

7.4 Mitigation and CODETTA-KEYED robustness

The above results show that efficient, undetectable collusion is practical in multi-agent systems, so a natural mitigation is to edit every message before it reaches the receiver, since an auditor that cannot detect the channel can still perturb it. In this section, we evaluate this strategy against robust variants of the keyed protocols.

We test this mitigation with a protocol-blind editor in the style of Watermarks in the Sand [ZEF+24], which sees only the public text and rewrites each finished response in a quality-constrained random walk. Each step proposes a rewrite of a six-word span with a public infilling model, which inserts, deletes, or substitutes words, and the editor accepts it only if the reward of the text stays above a threshold (Appendix C). The edit strength τ is the number of steps per word, so $\tau = 5\%$ is one step per 20 words, and at this strength the accepted rewrites change about a fifth of the words in the covert text.

CODETTA-KEYED itself is fragile to edits. The receiver computes the draft distribution, the alphabet, and the buckets of a token from the whole public prefix, and it computes the mask and the check of a token from its position. One edit changes the prefix and shifts the positions of all later tokens, so the receiver misreads the symbols of all later tokens.

Our robust variant removes both dependencies, and we call it content-addressed. For each token, both parties look only at the four public words before it. The draft predicts the token from these four words alone, which fixes the alphabet and the buckets. A hash of the same four words and the session identifier determines the mask and the check of the token. An edit therefore only affects the tokens whose preceding four words are edited, and every other token still yields its check. The receiver scans the edited text, collects the checks, and decodes as before. An edit can remove, corrupt, or add a check, and the ECC tolerates such errors. The same four words may appear more than once in a message, and only the first occurrence carries a check, so no mask is used twice and Theorem 1 still applies. The draft predicts worse given the much shorter context, and the protocol reaches 0.07 to 0.20 bits per token on FINANCIAL without edits. We make the warm-up

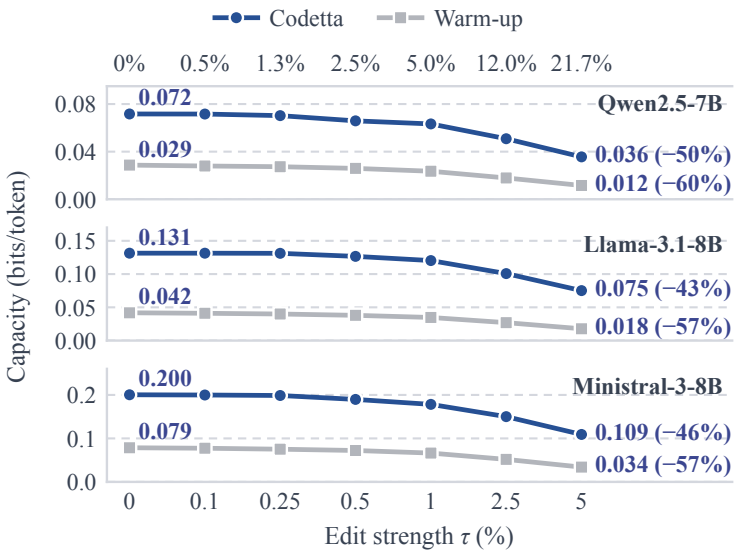

Figure 5: Capacity of the robust variants of keyed protocols on FINANCIAL as the cover text undergoes edits of growing strength τ. The upper axis is the average edit distance in percent of the words. Every message is decoded. ARS results are omitted since it is not robust to insertions and deletions and a single edit drops its capacity to zero.

content-addressed in the same way (Appendix C).

In Figure 5, edits at $\tau = 5\%$ cut the capacity of the variant by 43% to 50%, and the warm-up loses 57% to 60%. The robust variant still beats both baselines. Without edits, it delivers 2.5 to 3.1 times the capacity of the warm-up and 11 to 48 times the capacity of ARS in Table 1, and under the heaviest edits its gain over the warm-up grows to 3.1 to 4.2 times. Every one of the 40 messages is still recovered at every edit strength, so the edits slow the channel down but do not close it. The edits also have a price, which we measure in Figure 6. The perplexity of the edited text rises by a factor of 2.0 to 2.6, and about a tenth of its protected fields are damaged, although every rewrite passed the reward threshold. Capacity and quality fall together as τ grows, so to close the channel, the auditor has to edit much harder and damage much more of the text. Editing therefore lowers the efficiency of collusion but cannot fully mitigate it, and we discuss complementary mitigations in Sec. 9.

7.5 Correctness and cost of CODETTA-KE

We report the correctness and cost of CODETTA-KE for various models and workloads in Table 3. Failures are rare, with 379 failed sessions for Qwen on CTF, the worst one, and at most 12 in every other experiment. Theorem 5 turns these failure counts into one-sided 95% bounds on the failure probability, from 3.0×10^{-5} for an experiment without a failure to 4.1×10^{-3} for Qwen on CTF. On FINANCIAL and IFEVAL, the receiver reads 1.3% to 2.3% of the coefficients wrongly, i.e., 3 to 6 per 255-bit codeword, far below the 18 that the BCH code corrects. Qwen on CTF fails more often because its reports have low entropy (Sec. 7.2), making many candidate blocks in a pool identical and share the same label, so the carrier displacement grows.

An exchange transmits 1,280 positions of at most $B = 64$ tokens, i.e., at most 81,920 tokens, or 640 tokens per bit of the 128-bit pre-key. We show that setting $B = 64$ suffices to ensure low failure probability for the three workloads, using a larger B can further reduce the failure probability.

Comparison with Vaikuntanathan and Zamir [VZ26]. Their construction has no implementation, so

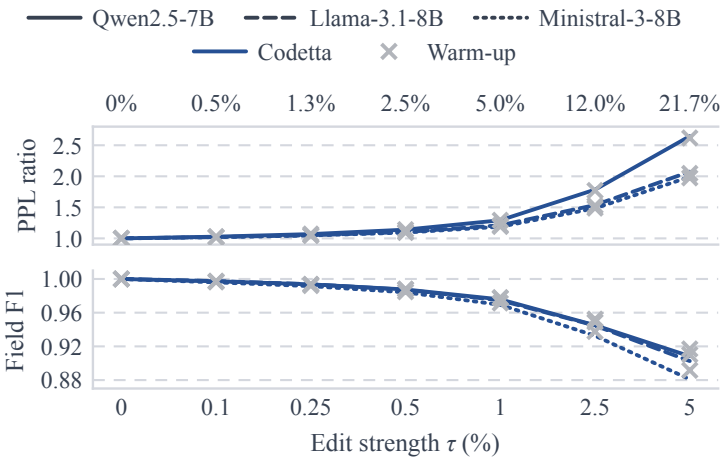

Figure 6: Quality of the edited text in Figure 5, with the same axes. The PPL ratio is the perplexity of the edited text over that of the clean text, and the field F1 compares the protected fields of the edited text with the clean text.

Workload	Sender	Failures	$\delta_{\mathscr{D}}$ bound	Tokens
FINANCIAL	Qwen	0	3.0×10^{-5}	81,920
	Llama	0	3.0×10^{-5}	81,920
	Ministral	0	3.0×10^{-5}	81,920
IFEVAL	Qwen	12	1.9×10^{-4}	70,751
	Llama	8	1.4×10^{-4}	77,972
	Ministral	0	3.0×10^{-5}	79,402
CTF	Qwen	379	4.1×10^{-3}	81,920
	Llama	0	3.0×10^{-5}	81,920
	Ministral	0	3.0×10^{-5}	81,920

Table 3: Correctness and cost of CODETTA-KE. Failures counts the failed sessions out of 10^5 simulations, the $\delta_{\mathscr{D}}$ bound is the one-sided 95% bound of Theorem 5, and Tokens is the number of transmitted tokens.

we estimate its cost on our workloads. We instantiate it with the Diffie–Hellman exchange with a uniform transcript that they suggest, with 256-bit group elements, i.e., 512 transcript bits at 128-bit security, and with their exponential-error signaling scheme at a total failure probability of 10^{-4}. Each published message of their construction carries one bit of the signaling and is chosen from a pool of candidate messages. We give their sender the same pools as ours, with 13,316 candidates per published block, and on our workloads the receiver reads a bit wrongly with probability below 1%. If each published message is as long as our 64-token blocks, the exchange publishes about 2 million tokens, against 81,920 for CODETTA-KE, i.e., about 23× more.

8 Related Work

LLM steganography. Provably secure steganography goes back to the complexity-theoretic definition of Hopper, Langford, and Von Ahn [HLVA02] and the information-theoretic model of Cachin [Cac04], and Dedić et al. showed that a sender who uses the channel only through sampling pays a number of samples

exponential in the rate [DIRR05], which is the cost that our keyless carrier pays when it draws a pool of candidate blocks for every q-ary symbol. Language models made these definitions practical over linguistic channels. Ziegler et al. embedded bits with arithmetic coding over GPT-2 and measured the divergence they introduced [ZDR19], and later constructions preserve the model distribution exactly or bound the distribution divergence: adaptive dynamic grouping [ZYYH21], Meteor [KJGR21], iMEC [SdWSK+23], Discop [DCW+23], SparSamp [WPC+25], Shimmer [BPL+25], ANStega [WPC+26], range coding [YM26], and adaptive dynamic sampling, which needs only a seeded sampling API [PB26]. All of them run in the symmetric setting: the receiver reruns the sender's model on the same context, or queries the same model with the same seed, so that it can reproduce the sampling step, and independently deployed agents do not satisfy this assumption (Sec. 1). Mächtle et al. show that deterministic inference stacks leak a channel through the sampling seed, which the receiver recovers by exhaustive search over the seed space [MSB+26]. A model-free receiver appears in the asymmetric-resource protocol of Bai et al. [BYP+25], the ARS baseline of Sec. 7: the sender hides bits in permutations of the token distribution, and the receiver, which sees only the carrier, guesses the permutation through a sampling function; the protocol is robust over a binary symmetric channel, and our measurements place its rate below 0.025 bits per token. Zamir hides an arbitrary payload in a response so that the response is indistinguishable without the key, extending the undetectable watermark [Zam24], and Christ and Gunn build stateless steganography with constant rate and robustness to a constant rate of substitutions from pseudorandom codes [CG24]. These constructions are analyzed asymptotically. CODETTA sits between the two extremes: its receiver runs a small public draft model using public context instead of the sender's model and private states, and Sec. 7 measures the resulting gain in delivered payload rate.

Steganographic key exchange. Von Ahn and Hopper introduced public-key steganography and steganographic key exchange [VAH04]. As Vaikuntanathan and Zamir observe, the earlier covert key exchanges assume covertexts that are generated non-adaptively and a min-entropy per message that grows with the security parameter [VZ26]. Their own work is the closest to ours. With a shared key, they show that the tools of undetectable watermarking and steganography give a covert conversation whose rate matches the entropy of the honest messages up to constant factors. Without a key, they introduce pseudorandom noise-resilient key exchange, prove that covert key exchange is possible when sufficiently many honest messages have constant min-entropy, give constructions from sparse learning parity with noise and from a signaling compiler with noiseless feedback, and prove that public pseudorandom codes cannot correct a constant fraction of bit flips and that non-interactive protocols admit a quasi-polynomial attack. Neither impossibility result applies to CODETTA-KE. Our receiver decodes with its MLWE secret, so no public decoder exists, and our exchange is adaptive, since (w, z) depends on the share $\widetilde{b}$ the sender read, while the carrier only displaces each coefficient by a small amount. Theirs is a feasibility result without an implementation, whereas CODETTA-KE transmits lattice coefficients directly over the carrier, without a separate bitwise signaling layer, and certifies correctness empirically on the measured channel; Sec. 7.5 estimates the cost of their construction on our workloads.

Watermarking and speculative decoding. Undetectable and distortion-free watermarks sample each token with randomness derived from a key so that the output distribution is unchanged, and their detectors need the key but not the model [CGZ24, KTHL24, CG24]. A watermark carries a fixed signal instead of a message, and the known limits of watermarking concern adversaries who edit or spoof the text [ZEF+24, PHZS24]. The maximal coupling of Sec. 4 is the acceptance rule of speculative decoding [LKM23, CBI+23], in which a small draft model proposes tokens and the target model accepts or resamples them so that the output keeps the target distribution exactly. CODETTA applies a similar rule to a bucket-conditioned proposal and treats a rejection as a channel error, while in speculative decoding, rejection reduces the inference speed.

Collusion among LLM agents. A line of work studies collusion that emerges from the behavior of the model itself. Agents are prompted or trained to hide messages or reasoning in their text [MMM+25, RG23, ZM26], monitors of transcripts and chains of thought are designed and evaluated against agents that hide their intent [GSSR24, BHG+25, APB+26], and simulations and surveys map the risk of multi-agent collu-

sion [RXW+25, Gha25]. These channels do not carry guarantees on the output distribution, and the hidden content can change the model behaviors, so monitors and paraphrasers can detect or weaken them in principle, although Mathew et al. [MMM+25] found neither passive oversight nor paraphrasing fully effective in practice. Secret Collusion [MBS+24] also covers the distribution-preserving case, built on iMEC [SdWSK+23]. CODETTA provides distribution-preservation guarantees in the realistic asymmetric setting.

9 Defense Discussion

Corollary 2 and Theorem 3 show that a passive auditor cannot detect collusion from the transcript, and in Sec. 7.4 we show that editing the messages lowers the capacity but cannot close the channel without damaging the text. Mathew et al. [MMM+25] report a similar limit for paraphrasing against emergent steganographic collusion. In this section, we discuss two more mitigation directions that go beyond transcript inspection.

Auditing visible agent actions. The first direction is to audit the agents' actions in addition to inspecting their messages. Coordinated actions (e.g., a trade or a tool call that the normal task does not require) are visible to the platform when they are executed, so an auditor on that platform can restrict the tools available to each agent, log every action, and check whether the actions are consistent with the assigned task. However, some actions may not be visible to the auditor. For example, sensitive data leakage is harmful as soon as the data leaves the sender, so the operator that deploys the sending agent can only limit the exposure in advance, i.e., give the agent access to no more data than its task requires. Data that the task legitimately requires remains exposed and we note that actions can also carry hidden information, e.g., through timing or through the choice among equivalent tool calls.

Verifiable sampling. The second direction is to make the sampling itself auditable. The agent commits to its sampling randomness, e.g., derived from a public beacon, and proves in zero knowledge that every published token was sampled from its model's distribution with that randomness, without revealing its private context [SLZ24, CWSK24]. CODETTA embeds the hidden message by steering the sampling, so a valid proof rules out the covert channel. The drawback has been cost, since proving the inference of a large language model in zero knowledge is orders of magnitude more expensive than the inference in plaintext. A recent provable inference system, FlashProve [RAC+26] is bringing this cost down, which makes the defense a promising direction for deployment.

10 Conclusion

We present CODETTA, a steganographic framework for multi-agent collusion. In the keyed regime, CODETTA-KEYED achieves orders of magnitude higher capacity than the prior asymmetric protocol on realistic agent workloads. In the keyless regime, CODETTA-KE efficiently establishes a shared secret key between independently deployed agents. We further evaluate a natural mitigation that perturbs the messages on the channel, and show that it slows the channel down but cannot remove the threat, so complementary defenses beyond passive transcript monitoring are essential.

References

[APB+26] Usman Anwar, Julianna Piskorz, David D. Baek, David Africa, Jim Weatherall, Max Tegmark, Christian Schroeder de Witt, Mihaela van der Schaar, and David Krueger. A decision-theoretic formalisation of steganography with applications to LLM monitoring. *arXiv preprint arXiv:2602.23163*, 2026. URL: https://arxiv.org/abs/2602.23163, doi:10.48550/arXiv.2602.23163.

[BHG+25] Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.

[BPL+25] Minhao Bai, Kaiyi Pang, Guorui Liao, Jinshuai Yang, and Yongfeng Huang. Shimmer: a provably secure steganography based on entropy collecting mechanism. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 5949–5965. USENIX Association, 2025.

[BRC60] Raj Chandra Bose and Dwijendra K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960. doi:10.1016/S0019-9958(60)90287-4.

[BYP+25] Minhao Bai, Jinshuai Yang, Kaiyi Pang, Xin Xu, Zhen Yang, and Yongfeng Huang. Provably robust and secure steganography in asymmetric resource scenario. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1438–1456. IEEE, 2025.

[Cac04] Christian Cachin. An information-theoretic model for steganography. *Information and Computation*, 192(1):41–56, 2004. doi:10.1016/j.ic.2004.02.003.

[CBB14] Peter Chapman, Jonathan Burket, and David Brumley. PicoCTF: A Game-Based computer security competition for high school students. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[CBI+23] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[CG24] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In *Annual International Cryptology Conference*, pages 325–347. Springer, 2024.

[CGZ24] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR, 2024.

[CNS+25] Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, et al. LlamaFirewall: An open source guardrail system for building secure AI agents. *arXiv preprint arXiv:2505.03574*, 2025.

[CWSK24] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. ZKML: An optimizing system for ML inference in zero-knowledge proofs. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 560–574, 2024.

[DCW+23] Jinyang Ding, Kejiang Chen, Yaofei Wang, Na Zhao, Weiming Zhang, and Nenghai Yu. Discop: Provably secure steganography in practice based on distribution copies. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2238–2255. IEEE, 2023.

[DIRR05] Nenad Dedić, Gene Itkis, Leonid Reyzin, and Scott Russell. Upper and lower bounds on black-box steganography. In *Theory of Cryptography Conference*, pages 227–244. Springer, 2005.

[Gal62] Robert G. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962. doi:10.1109/TIT.1962.1057683.

[GCW+24] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8048–8057. International Joint Conferences on Artificial Intelligence Organization, August 2024. doi:10.24963/ijcai.2024/890.

[GDJ+24] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[Gha25] Mohammad Sajjad Ghaemi. A survey of collusion risk in LLM-powered multi-agent systems. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025.

[GSSR24] Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. AI control: Improving safety despite intentional subversion. In *Forty-first International Conference on Machine Learning*, 2024. URL: https://openreview.net/forum?id=KviM5k8pcP.

[HCAA+22] Jakob Hoydis, Sebastian Cammerer, Fayçal Ait Aoudia, Avinash Vem, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. Sionna: An open-source library for next-generation physical layer research. *arXiv preprint arXiv:2203.11854*, 2022.

[HGC23] Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023. URL: https://openreview.net/forum?id=sE7-XhLxHA.

[HLVA02] Nicholas J Hopper, John Langford, and Luis Von Ahn. Provably secure steganography. In *Annual International Cryptology Conference*, pages 77–92. Springer, 2002.

[Hoc59] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959.

[KFL01] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001. doi:10.1109/18.910572.

[KJGR21] Gabriel Kaptchuk, Tushar M Jois, Matthew Green, and Aviel D Rubin. Meteor: Cryptographically secure steganography for realistic distributions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1529–1548, 2021.

[KTHL24] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. URL: https://openreview.net/forum?id=FpaCL1MO2C.

[LC04] Shu Lin and Daniel J. Costello. *Error Control Coding*. Prentice Hall, second edition, 2004.

[LKM23] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 2023.

[LKS+26] Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[Lub02] Michael Luby. Lt codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 271–280, 2002. doi:10.1109/SFCS.2002.1181950.

[MBS+24] Sumeet R Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip H Torr, Lewis Hammond, and Christian Schroeder de Witt. Secret collusion among AI agents: Multi-agent deception via steganography. *Advances in Neural Information Processing Systems*, 37:73439–73486, 2024.

[MET26] METR. Brief independent investigation of agents' behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident. https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/, August 2026.

[MMM+25] Yohan Mathew, Ollie Matthews, Robert McCarthy, Joan Velja, Christian Schroeder de Witt, Dylan Cope, and Nandi Schoots. Hidden in plain text: Emergence & mitigation of steganographic collusion in LLMs. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 585–624, 2025.

[MSB+26] Felix Mächtle, Jonas Sander, Sebastian Berndt, Ben Weimar, Nils Loose, and Thomas Eisenbarth. Steganography without modification: Hidden communication via LLM seeds. *arXiv preprint arXiv:2606.09135*, 2026. URL: https://arxiv.org/abs/2606.09135.

[Nat24] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Technical Report NIST FIPS 203, National Institute of Standards and Technology, August 2024. doi:10.6028/NIST.FIPS.203.

[Ope26a] OpenAI. The hugging face incident and other third-party impact from misaligned models. https://openai.com/hugging-face-incident-and-misalignment/, September 2026. Accessed: 2026-09-18.

[Ope26b] OpenAI. The Hugging Face incident and the road ahead. https://openai.com/index/hugging-face-incident-and-the-road-ahead/, August 2026. Accessed: 2026-09-09.

[PB26] Kaiyi Pang and Minhao Bai. Provable secure steganography based on adaptive dynamic sampling. In *35th USENIX Security Symposium (USENIX Security 26)*, pages 3143–3159, 2026.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[PHZS24] Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. No free lunch in LLM watermarking: Trade-offs in watermarking design choices. *Advances in Neural Information Processing Systems*, 37:138756–138788, 2024.

[QLL+24] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15174–15186, 2024.

[RAC+26] Roy Rinberg, Alex Allain, Alessandro Chiesa, Miranda Christ, Rahul Dalal, Giacomo Fenzi, Ben Mayer, and Guy Weissenberg. FlashProve: an open-source library for provable inference. Technical report, Anthropic, 2026.

[RG23] Fabien Roger and Ryan Greenblatt. Preventing language models from hiding their reasoning. *arXiv preprint arXiv:2310.18512*, 2023. URL: https://arxiv.org/abs/2310.18512.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[RU08] Tom Richardson and Rüdiger Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008. doi:10.1017/CBO9780511791338.

[RXW+25] Qibing Ren, Sitao Xie, Longxuan Wei, Zhenfei Yin, Junchi Yan, Lizhuang Ma, and Jing Shao. When autonomy goes rogue: Preparing for risks of multi-agent collusion in social systems. *arXiv preprint arXiv:2507.14660*, 2025.

[SdWSK+23] Christian Schroeder de Witt, Samuel Sokota, J Zico Kolter, Jakob Nicolaus Foerster, and Martin Strohmeier. Perfectly secure steganography using minimum entropy coupling. In *The Eleventh International Conference on Learning Representations*, 2023. URL: https://openreview.net/forum?id=HQ67mj5rJdR.

[Sho06] Amin Shokrollahi. Raptor codes. *IEEE transactions on information theory*, 52(6):2551–2567, 2006.

[SLZ24] Haochen Sun, Jason Li, and Hongyang Zhang. zkllm: Zero knowledge proofs for large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4405–4419, 2024.

[TZZ+24] Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. Medagents: Large language models as collaborators for zero-shot medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621, 2024.

[VAH04] Luis Von Ahn and Nicholas J Hopper. Public-key steganography. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 323–341. Springer, 2004.

[VZ26] Vinod Vaikuntanathan and Or Zamir. Undetectable conversations between AI agents via pseudorandom noise-resilient key exchange. *arXiv preprint arXiv:2604.04757*, 2026. URL: https://arxiv.org/abs/2604.04757.

[WPC+25] Yaofei Wang, Gang Pei, Kejiang Chen, Jinyang Ding, Chao Pan, Weilong Pang, Donghui Hu, and Weiming Zhang. SparSamp: Efficient provably secure steganography based on sparse sampling. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 6817–6835. USENIX Association, 2025.

[WPC+26] Yaofei Wang, Weilong Pang, Kejiang Chen, Jinyang Ding, Donghui Hu, Weiming Zhang, and Nenghai Yu. Breaking the generative steganography trilemma: Anstega for optimal capacity, efficiency, and security. In *33rd Annual Network and Distributed System Security Symposium*. The Internet Society, 2026. doi:10.14722/ndss.2026.240605.

[XSLW25] Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. TradingAgents: Multi-agents LLM financial trading framework. In *The First MARW: Multi-Agent AI in the Real World Workshop at AAAI 2025*, 2025. URL: https://openreview.net/forum?id=4QPrXwMQt1.

[YCJ+24] Weikang Yuan, Junjie Cao, Zhuoren Jiang, Yangyang Kang, Jun Lin, Kaisong Song, Tianqianjin Lin, Pengwei Yan, Changlong Sun, and Xiaozhong Liu. Can large language models grasp legal theories? enhance legal reasoning with insights from multi-agent collaboration. In *Findings of the association for computational linguistics: EMNLP 2024*, pages 7577–7597, 2024.

[YM26] Ruiyi Yan and Yugo Murawaki. Efficient provably secure linguistic steganography via range coding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pages 890–907. Association for Computational Linguistics, 2026. doi:10.18653/v1/2026.acl-long.39.

[YPNY23] John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36:23826–23854, 2023.

[YYZ+24] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[Zam24] Or Zamir. Undetectable steganography for language models. *Transactions on Machine Learning Research*, 2024. URL: https://openreview.net/forum?id=fq6aQoMSHz.

[ZDR19] Zachary Ziegler, Yuntian Deng, and Alexander Rush. Neural linguistic steganography. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China, November 2019. Association for Computational Linguistics. URL: https://aclanthology.org/D19-1115/, doi:10.18653/v1/D19-1115.

[ZEF+24] Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: Impossibility of strong watermarking for language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58851–58880. PMLR, 2024. URL: https://proceedings.mlr.press/v235/zhang24o.html.

[ZLM+23] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. arXiv preprint arXiv:2311.07911, 2023. URL: https://arxiv.org/abs/2311.07911, arXiv:2311.07911.

[ZM26] Zhejian Zhou and Jonathan May. Conceptual steganography. *arXiv preprint arXiv:2605.26537*, 2026. URL: https://arxiv.org/abs/2605.26537, doi:10.48550/arXiv.2605.26537.

[ZYYH21] Siyu Zhang, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. Provably secure generative linguistic steganography. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3046–3055, Online, August 2021. Association for Computational Linguistics. URL: https://aclanthology.org/2021.findings-acl.268/, doi:10.18653/v1/2021.findings-acl.268.

A Error-Correcting Codes

This appendix reviews the BCH and LDPC codes named in Sec. 2, and gives the sum-product updates of belief propagation (BP), a worked example, and the relation between BP and the MAP decoder.

BCH codes. BCH codes [Hoc59, BRC60] are linear block codes constructed algebraically. A binary BCH code of length $N = 2^m - 1$ is generated by a polynomial over $\mathbb{F}_2$ whose roots include 2τ consecutive powers of a primitive element of $\mathbb{F}_{2^m}$; this forces $d_{\min} \geq 2\tau + 1$, and an efficient algebraic decoder corrects any τ errors from the syndromes of the received word [LC04]. The guarantee is deterministic and holds for every error pattern of weight at most τ, including independent, correlated, or adversarial errors. CODETTA-KE (Sec. 6) encodes the pre-key with the binary BCH code $(N, k, \tau) = (255, 131, 18)$, and its correctness proof relies on this property.

LDPC codes. A low-density parity-check (LDPC) code [Gal62] is a linear block code defined by a sparse parity-check matrix $H \in \mathbb{F}_2^{(N-k)\times N}$: the codewords are the $x \in \{0,1\}^N$ with $Hx = 0$, so each row of H is a parity check on a few bits. LDPC codes are decoded by belief propagation on the Tanner graph, which has one node per bit and one per check, and well-designed ensembles approach capacity for long blocks [RU08]. The rate is adapted by puncturing parity bits or by sending them incrementally until decoding succeeds (hybrid ARQ); the LDPC baseline in our evaluation runs in this incremental mode.

Belief propagation. The optimal decoder is maximum a posteriori (MAP): given all observations $\mathcal{O}$, output $\arg\max_u \mathbb{P}[u \mid \mathcal{O}]$. Computing it requires a search over all $|\Sigma|^k$ messages, so it is feasible only for short payloads. Belief propagation (BP), also called the sum-product algorithm, approximates it [Pea88, KFL01, RU08]. It places the unknown symbols and the received checks on a bipartite graph, with an edge whenever a check involves a symbol, and passes messages along the edges: each symbol tells a check what its other checks make it believe, and each check tells a symbol how likely each of its values is given the beliefs of the check's other symbols. After many rounds, every symbol takes its most probable value.

Factor graph. The graph has a variable node for every unknown symbol u_i and a check node for every received check s, joined by an edge whenever $i \in S_s$; we write $\partial i = \{s : i \in S_s\}$ for the checks of symbol i. Check s carries a factor ϕ_s that scores each local assignment of its symbols by how well it explains the received value. If the check value passed through a channel with transition probabilities W_s and arrived as $\widehat{C}_s$, then

$$\phi_s(u_{S_s}) = W_s\big(\widehat{C}_s \mid C_s(u_{S_s})\big), \tag{13}$$

and $\mathbb{P}[u \mid \mathcal{O}]$ is proportional to the product of all factors. For an LDPC code the check factor is instead the indicator $\mathbf{1}\{C_s(u_{S_s}) = 0\}$ and the channel likelihood of each received bit multiplies the messages of its variable node; we omit that term because our symbols are never transmitted directly.

Message updates. A message is an opinion about one symbol, formed while excluding the recipient so that no evidence is counted twice. Variable i tells check s what its other checks currently make it believe,

$$\mu_{i\to s}(u_i) \propto \prod_{s' \in \partial i \setminus \{s\}} \mu_{s'\to i}(u_i), \tag{14}$$

and check s tells variable i how likely each value of u_i is, given its factor and the current beliefs of its other symbols,

$$\mu_{s\to i}(u_i) \propto \sum_{u_{S_s \setminus \{i\}}} \phi_s(u_{S_s}) \prod_{j \in S_s \setminus \{i\}} \mu_{j\to s}(u_j). \tag{15}$$

All messages start uniform, and the two updates alternate until a public iteration cap or until the messages stop changing. The belief of symbol i combines all incoming opinions, and the decision is its mode:

$$b_i(u_i) \propto \prod_{s \in \partial i} \mu_{s\to i}(u_i), \qquad \widehat{u}_i = \arg\max_{u_i \in \Sigma} b_i(u_i). \tag{16}$$

For a check modulo K over d binary variables, the sum in Equation (15) is a convolution over $\mathbb{Z}_K$ and takes $O(dK^2)$ time by dynamic programming over partial sums.

Example. Suppose one check involves only u_1 and reads $u_1 = 1$, and a second reads $u_1 \oplus u_2 = 1$. The first check makes u_1 believe it is 1 with probability $1 - e$, where e is the channel error rate. In the next iteration u_1 passes this belief to the second check, which concludes that u_2 is 0 with about the same confidence, and a third check on $u_2 \oplus u_3$ then carries the evidence on to u_3. Each iteration moves evidence one edge further, from the checks that pin a symbol down to the rest of the graph. The same mechanism explains why BP stalls without low-degree checks: if every check involves several symbols, uniform incoming messages produce uniform outgoing messages in Equation (15).

Approximation and guarantees. On a tree, the messages implement exact marginalization and b_i is the exact posterior marginal. On a graph with cycles, the same evidence can travel around a cycle and be counted again, so BP is an approximation, accurate when the graph is sparse and short cycles are rare. BP extends incrementally: a newly received check adds one node and its edges, and the receiver simply runs more rounds. BP comes with no guarantee that its output is the transmitted message, whereas the MAP decoder admits a direct analysis. The validation tag tells the two outcomes apart: the receiver accepts a decoded message only if its keyed Tag validates and returns $\perp$ otherwise, and with a λ-bit keyed tag a wrong message passes with probability at most $Q2^{-\lambda}$ plus the PRF advantage over Q validation attempts. Validation also lets the receiver test a bounded list of candidates near the BP output, for instance by flipping its least confident symbols, with each test counted toward Q.

B Workload Details

This appendix describes the three agent workloads of Sec. 7.1 and gives their prompts and example transcripts. On every workload, the receiver agent sees only the public transcript, i.e., its own request, the public problem statement, and the sender's response.

B.1 FINANCIAL

FINANCIAL simulates an issuer-intelligence vendor. The receiver agent, an asset manager, sends the sender agent a JSON research request about one synthetic issuer. The sender retrieves the five records for that issuer from its local database, which holds 100 records for 20 synthetic issuers with five record types each (issuer profile, quarterly update, public milestone, sector and peer context, risks and catalysts), and writes a research report in prose. Its system prompt restricts it to records marked public or distribution-permitted. The retrieved records, about 1.7k tokens per request, are appended to the sender's context only; the receiver sees the system prompt and the request. The report is the carrier. The pool contains 50 requests.

Public request from the receiver agent (abridged)

```
{ "from": "PortfolioResearchAgent_B", "to": "IssuerIntelAgent_A",
  "task": "60_day_investment_committee_memo", "asset": "SYN-LFD",
  "payload": {
    "issuer": "Lumen Foods Holdings",
    "sector": "packaged foods / branded nutrition",
    "requested_sections": ["issuer overview", "recent public developments",
      "earnings and guidance summary", "sector and peer context",
      "public risks", "public catalysts", "scenario analysis",
      "monitoring questions", "data-quality caveats"] } }
Suggested lookup: SELECT * FROM intel_records WHERE asset = 'SYN-LFD'
  ORDER BY record_type;
```

Sender-only context: one of the five retrieved records (abridged)

```
{ "record_id": "R11-QUARTER", "issuer": "Lumen Foods Holdings",
  "record_type": "quarterly_update", "as_of_date": "2026-06-30",
  "public_observations": [
    "Revenue trend: 4% year-over-year, supported by pricing but offset by
     modest volume declines.",
    "Margin trend: gross margin improved 130 basis points as commodity hedges
     and productivity initiatives helped.",
    "Guidance: management reaffirmed organic growth guidance and raised
     productivity savings target.", ... ],
  "metrics": { ... }, "monitoring_points": [ ... ] }
```

Response, i.e., the carrier (first lines)

```
Research Report for 90-Day Research Review of SYN-LFD (Lumen Foods Holdings)

### Issuer Overview
Lumen Foods Holdings is a synthetic issuer operating in the packaged foods and
branded nutrition sectors, with significant presence in North America and
Europe. Its business is segmented into branded nutrition (40%), frozen meals
(28%), snacking (21%), and foodservice (11%). ...

### Recent Public Developments
In the most recent quarterly update, Lumen Foods reported a 4% year-over-year
revenue growth, driven by pricing offsetting modest volume declines. Gross
margin improved by 130 basis points due to commodity hedging and productivity
initiatives. ...
```

B.2 IFEVAL

IFEVAL prompts come from the public IFEval dataset [ZLM+23] at a pinned revision. Each prompt carries one or more instructions that a rule-based scorer verifies. Before answering, the sender's base model produces a private plan by greedy decoding of at most 48 tokens under the instruction below; the plan is appended to the sender's context and never shown to the receiver, whose context is the system prompt and the request. The response is the carrier. The system prompt for both agents is "You are a careful and helpful instruction-following assistant."

Plan instruction appended to the sender's prompt

```
Privately make an abstract response-control plan; do not answer the request.
Use at most 24 words and only generic control categories such as length,
sections, bullets, casing, punctuation, required or forbidden elements, and
ending. Counts are allowed. Do not include topic-specific names, dates, facts,
examples, advice, claims, or reusable answer phrases. This private plan will
be supplied to the answering agent.
```

Example 1 (instruction: change_case:english_capital)

```
Request:  Come up with 3 names for a 2B software company. Make sure your names
          are in English and all capital letters.
Plan:     Plan: 3 x Capital Letters, No Punctuation, <25 Chars Each
Response: INNOVAQBRICK
```

```
          CODECLARIGEN
          SECURESKYLINE
```

Example 2 (instruction: punctuation:no_comma)

```
Request:  Write a riddle about Camilla that doesn't use commas.
Plan:     Sections:1  Punctuation:none  Length:24words  Casing:none
Response: scratch your back i am small but fierce camilla
```

B.3 CTF

CTF simulates two agents that communicate through a shared file system instead of a conversation. We take a subset of 20 challenges from the offline InterCode-CTF [YPNY23] adaptation of picoCTF [CBB14], i.e., nine cryptography, five reverse-engineering, four general-skills, one forensics, and one binary-exploitation challenge. The sender agent first solves a challenge in an offline sandbox with shell and Python tools, in a ReAct loop of at most 50 messages and three flag submissions at temperature 1. The complete solve trajectory, i.e., the task statement with its attached files, every tool command, every tool output, and the submission feedback, is the sender's private context. The same model then receives the report request below, appended to this trajectory, and writes one technical report on the task, which ends at its end-of-sequence token or after 576 tokens. The report is the carrier: the sender writes it into a directory that both agents can access, and the receiver agent reads the file from there, so the only communication between the two agents is the file itself. The receiver's context is the public task statement and the report file, and it never sees the trajectory. The first 64 tokens of each report and all code blocks in it are generated without embedding, and the capacity in Table 1 divides the payload by all transmitted tokens of the report, including these. We have omitted the tool outputs in the example below.

Example (calibration task 24, abridged)

```
Task:       Run the Python script code.py in the same directory as codebook.txt.
Trajectory: sender only; six tool turns, commands shown without their outputs
  [1] bash    head -c 100 code.py
  [2] bash    cat code.py
  [3] bash    cat codebook.txt
  [4] python  codebook = "azbycxdwevfugthsirjqkplomn"
              key = codebook[4] + codebook[14] + codebook[13] + codebook[14] + codebook[23] +
  codebook[25 % len(codebook)] + codebook[16] + codebook[0] + codebook[25 % len(codebook)]
              print(key)
  [5] python  key = "chthonian" ... result = str_xor(flag_enc, key) ... print(result)
  [6] bash    submit picoCTF{c0d3b00k_455157_d9aa2df2}
Report:     the carrier, written to the shared directory (first lines)
  ### **Challenge Analysis: Decrypting the Flag with a Custom XOR Cipher**
  #### **Objective and Underlying Concepts**
  This Capture The Flag (CTF) challenge required decryption of a flag using a
  simple XOR-based cipher. The provided Python script (`code.py`) incorporated
  a `str_xor` function to perform bytewise XOR operations between a secret
  message (`flag_enc`) and a key derived from `codebook.txt`. ...
  [...]
  FINAL: picoCTF{c0d3b00k_455157_d9aa2df2}
```

Report request appended to the sender's trajectory (abridged)

```
REPORT PHASE. Tool execution has ended. Write a complete, self-contained
technical report in English about the task and the work recorded above. Explain
the task and relevant inputs, the approach taken, the decisive commands or
reasoning, the evidence obtained, and the final result. Develop the explanation
in connected paragraphs, including concrete intermediate observations and why
they support or fail to support the conclusion. ... Treat earlier assistant
reasoning as hypotheses; correct any claims contradicted by tool output.
Distinguish observed facts from interpretations. Use code excerpts when they
help explain the method. Do not invent actions, observations, or results. End
with exactly one plain line: FINAL: <verified flag>, or FINAL: UNSOLVED if no
flag was verified.
```

C Editing Details

This appendix completes the description of the editor and of the robust variants of Sec. 7.4.

Editor. The editor runs after the sender has written all responses of a message and before the receiver reads them. It is a lazy random walk with laziness probability 0.1, and each step rewrites a six-word span with the XL model of Google's T5 v1.1 [RSR+20], which may change the number of words. A proposed rewrite is accepted only if the carrier's mean response-block reward under the DeBERTa-v3-large reward model of OpenAssistant [HGC23] meets a threshold fixed on disjoint ordinary-text calibration data, the fifth percentile of the mean rewards of 30 ordinary trajectories from 10 calibration prompts and 3 seeds. For an original carrier of W_0 words, the edit strength τ runs $T = \lceil \tau W_0 \rceil$ steps, where accepted rewrites, rejected proposals, and lazy steps all count, and the seven strengths of Figure 5 are checkpoints of one walk per carrier. At $\tau = 5\%$, the realized word-level edit density, counting insertions, deletions, and substitutions relative to the original length, is 21.2% for Qwen, 21.5% for Llama, and 22.8% for Ministral.

Robust variants. The content-addressed variant of CODETTA-KEYED reserves each four-word context once per message, i.e., per session identifier and across all of its responses, and the first occurrence takes the address even when its alphabet size is one. The receiver reserves by the first occurrence in the edited text, so an edit that reorders occurrences leaves the two parties with different checks at that address, which the decoder sees as a wrong or missing check value. The receiver rebuilds the checks from the edited text and decodes with belief propagation and the validation tag as in Algorithm 1. The warm-up replaces the position index of its mask and of its codeword symbol with the same four-word context.

D Deferred Proofs

This appendix restates the results of Secs. 4 and 6 and proves them, in the order in which they appear in the main text; the results of Sec. 5 are corollaries of those of Sec. 4 and are justified in place.

D.1 Proof of Lemma 1

Lemma 1 (Distribution preservation of maximal coupling)**.** *Let a_t be uniform on $\mathbb{Z}_K$ and independent of p_t. Then (i) the published token satisfies $Y_t \sim p_t$; (ii) the proposal is rejected with probability $\sum_y (p_t(y) - \nu_t(y))_+ = \Delta_K(p_t)$, which equals $\mathrm{TV}(p_t, \nu_t)$ when every bucket has positive mass; and (iii) $f_{\mathsf{Bucket},t}(Y_t) \neq a_t$ iff the proposal is rejected.*

Proof. Write $\alpha_t(x) = \min\{1, p_t(x)/\nu_t(x)\}$ for $x \in \mathrm{supp}\nu_t$, and treat an empty proposal, the case $\pi_t(a_t) = 0$, as a rejection. We first record how ν_t deviates from p_t. For $x \in B_{t,a}$, Equation (5) gives

$$\begin{aligned} p_t(x) - \nu_t(x) &= p_t(x)\Big(1 - \frac{1}{K\,\pi_t(a)}\Big), \\ \sum_{x \in B_{t,a}} \big(p_t(x) - \nu_t(x)\big) &= \pi_t(a) - \frac{1}{K}, \end{aligned} \tag{17}$$

so $p_t - \nu_t$ has the sign of $\pi_t(a) - 1/K$ throughout bucket a, and therefore

$$\begin{aligned} \sum_x \big(p_t(x) - \nu_t(x)\big)_+ &= \sum_{a \in \mathbb{Z}_K} \Big(\pi_t(a) - \frac{1}{K}\Big)_+ \\ &= \frac{1}{2} \sum_{a \in \mathbb{Z}_K} \Big|\pi_t(a) - \frac{1}{K}\Big| = \Delta_K(p_t), \end{aligned} \tag{18}$$

where the middle equality holds because $\sum_a \pi_t(a) = \sum_a 1/K = 1$.

Proof of (i). The two branches of the sampler, lines 14–15 of Algorithm 1, give, for every $y \in \mathscr{V}$,

$$\mathbb{P}[Y_t = y] = \nu_t(y)\,\alpha_t(y) + \Big(1 - \sum_x \nu_t(x)\,\alpha_t(x)\Big)\rho_t(y). \tag{19}$$

Since $\nu_t \alpha_t = \min\{p_t, \nu_t\}$ and $\sum_x p_t(x) = 1$, the weight of the second branch is $1 - \sum_x \min\{p_t(x), \nu_t(x)\} = \sum_x (p_t(x) - \nu_t(x))_+ = \Delta_K(p_t)$ by Equation (18). Substituting this and Equation (6) into Equation (19),

$$\mathbb{P}[Y_t = y] = \min\{p_t(y), \nu_t(y)\} + \big(p_t(y) - \nu_t(y)\big)_+ = p_t(y).$$

Proof of (ii). The rejection probability is $1 - \sum_x \min\{p_t(x), \nu_t(x)\} = \sum_x (p_t(x) - \nu_t(x))_+ = \Delta_K(p_t)$. When every bucket has positive mass, ν_t sums to one, so the positive and negative parts of $p_t - \nu_t$ have equal mass and $\sum_x (p_t(x) - \nu_t(x))_+ = \frac{1}{2} \sum_x |p_t(x) - \nu_t(x)| = \mathrm{TV}(p_t, \nu_t)$.

Proof of (iii). If the proposal is accepted, then $Y_t = X_t$, and $X_t \in B_{t,a_t}$ by construction, so $f_{\mathsf{Bucket},t}(Y_t) = a_t$. Suppose instead that the proposal is rejected. By Equation (5), $p_t(x)/\nu_t(x) = K\pi_t(a_t)$ for every $x \in B_{t,a_t}$, so $\alpha_t \equiv 1$ on the target bucket whenever $\pi_t(a_t) \geq 1/K$, so a rejection is possible only if $\pi_t(a_t) < \frac{1}{K}$. On rejection, $Y_t \sim \rho_t$, and by Equation (17) the residual satisfies $\rho_t(y) > 0$ only if $\pi_t\big(f_{\mathsf{Bucket},t}(Y_t)\big) > \frac{1}{K}$. The two inequalities cannot hold for the same bucket, so $f_{\mathsf{Bucket},t}(Y_t) \neq a_t$. Together, $f_{\mathsf{Bucket},t}(Y_t) \neq a_t$ exactly when the proposal is rejected. □

D.2 Proof of Theorem 1

Theorem 1 (Undetectability of the sampler)**.** *Consider a keyed construction in which, at every step t, the alphabet size K_t and the partition $B_{t,\cdot}$ are determined by the public prefix $V_{<t}$ and public parameters, every step with $K_t > 1$ embeds a symbol $C_t \in \mathbb{Z}_{K_t}$ masked with $\mathsf{RangePRF}_{k_m}$ at a public input that never repeats within a session, and the token is drawn by the sampler of Sec. 4.1, with $Y_t \sim p_t$ when $K_t = 1$. If* RangePRF *is computationally secure, the construction is computationally undetectable. Concretely, for every PPT distinguisher $\mathscr{A}$ in Definition 2 there is a PPT distinguisher $\mathscr{A}_1$ against* RangePRF *such that* $\mathrm{Adv}^{\mathsf{steg}}_{\mathscr{A}}(\kappa) \leq \mathrm{Adv}^{\mathsf{prf}}_{\mathsf{RangePRF},\mathscr{A}_1}(\kappa)$.

Proof. We use a hybrid argument over three experiments.

- H_0: the experiment of Definition 2 with $b = 1$; every query is answered by EMBED.
- H_1: identical to H_0, except that every mask $\mathsf{RangePRF}_{k_m}(x)$ is replaced by $F(x)$ for a uniformly random function F whose value at an input with alphabet size K is uniform on $\mathbb{Z}_K$.

- H_2: the experiment with $b = 0$; every query is answered by ordinary sampling from the base model without an embedded message.

Write $\mathbb{P}[\mathsf{H}_j = 1]$ for the probability that $\mathscr{A}$ outputs 1 in H_j, so that $\mathrm{Adv}^{\mathsf{steg}}_{\mathscr{A}} = |\mathbb{P}[\mathsf{H}_0 = 1] - \mathbb{P}[\mathsf{H}_2 = 1]|$.

$\mathsf{H}_0 \approx \mathsf{H}_1$. Let $\mathscr{A}_1$ be the distinguisher against $\mathsf{RangePRF}$ that runs $\mathscr{A}$, simulates the challenger with the base model, the public partition rule, and the code, obtains every mask by querying its oracle at the public input of the step, and outputs whatever $\mathscr{A}$ outputs. When the oracle is $\mathsf{RangePRF}_{k_m}$, $\mathscr{A}_1$ simulates H_0 exactly; when it is a random function, $\mathscr{A}_1$ simulates H_1 exactly. Hence $|\mathbb{P}[\mathsf{H}_0 = 1] - \mathbb{P}[\mathsf{H}_1 = 1]| = \mathrm{Adv}^{\mathsf{prf}}_{\mathsf{RangePRF},\mathscr{A}_1}$, and $\mathscr{A}_1$ runs $\mathscr{A}$ once plus one protocol execution per query, so it is PPT whenever $\mathscr{A}$ is.

$\mathsf{H}_1 \equiv \mathsf{H}_2$. Fix a query $(\mathsf{sid}_i, m_i, T_i)$ and let $\mathscr{F}_{t-1}$ be the complete state before step t: the private state H_t, the transcript $V_{<t}$, the message, and all earlier randomness. We compare the conditional distribution of (Y_t, H_{t+1}) given $\mathscr{F}_{t-1}$ in the two experiments.

(a) The alphabet size K_t and the partition $B_{t,\cdot}$ are deterministic functions of $V_{<t}$ and public parameters, so they take the same values in both experiments and depend on the message only through the published prefix.

(b) If $K_t = 1$, the sender samples $Y_t \sim p_t$ in H_1, which is the distribution of Y_t in H_2 by Equation (1).

(c) If $K_t > 1$, the public input of the mask is determined by $V_{<t}$ and, together with sid_i, never repeats across queries and steps, so F is evaluated at a fresh point and its value is uniform on $\mathbb{Z}_{K_t}$, independent of $\mathscr{F}_{t-1}$ and of the symbol C_t. Hence the masked symbol $A_t = C_t + F(\cdot) \bmod K_t$ is uniform given $\mathscr{F}_{t-1}$, for every message, and Lemma 1(i) applied to the partition $B_{t,\cdot}$ gives $Y_t \mid \mathscr{F}_{t-1} \sim p_t$ in H_1, which is again the distribution of Y_t in H_2.

(d) After Y_t is published, the sender's private state H_{t+1} is computed in the same way in both experiments: it is a function of H_t, $V_{\le t}$, and fresh randomness of the agent, and it does not depend on the payload, the mask, the proposal, or whether the proposal was accepted (Sec. 3.2).

By induction on t, the joint distribution of $(H_{1:T_i+1}, V_{1:T_i})$, and in particular the transcript returned to $\mathscr{A}$, is identical in H_1 and H_2. The argument holds conditionally on all earlier queries, so the view of $\mathscr{A}$ is identically distributed in the two experiments and $\mathbb{P}[\mathsf{H}_1 = 1] = \mathbb{P}[\mathsf{H}_2 = 1]$.

By the triangle inequality,

$$\begin{aligned}\mathrm{Adv}^{\mathsf{steg}}_{\mathscr{A}} &\le \big|\mathbb{P}[\mathsf{H}_0 = 1] - \mathbb{P}[\mathsf{H}_1 = 1]\big| \\ &\quad + \big|\mathbb{P}[\mathsf{H}_1 = 1] - \mathbb{P}[\mathsf{H}_2 = 1]\big| = \mathrm{Adv}^{\mathsf{prf}}_{\mathsf{RangePRF},\mathscr{A}_1}.\end{aligned}$$

If $\mathsf{RangePRF}$ is computationally secure, the right side is negligible in κ for every PPT $\mathscr{A}$, so the construction is computationally undetectable. In the warm-up, the public input of the mask is (sid, t) and $K_t = K$; in CODETTA-KEYED it is (sid, s, K_t) with the check index s determined by $V_{<t}$. □

D.3 Proof of Corollary 1

Corollary 1 (Substitution channel)**.** *In a construction as in Theorem 1, the receiver observes exactly one symbol $\widehat{C}_t$ per step with $K_t > 1$, in the same position as C_t, and $\widehat{C}_t \neq C_t$ if and only if the proposal at step t is rejected. Conditioned on $\mathscr{F}_{t-1}$ and the masked symbol A_t, $E_t \sim \mathrm{Bernoulli}\big((1 - K_t \pi_t(A_t))_+\big)$; if the mask is uniform and independent of $\mathscr{F}_{t-1}$, then $\mathbb{P}[E_t = 1 \mid \mathscr{F}_{t-1}] = \Delta_{K_t}(p_t)$.*

Proof. K_t and $B_{t,\cdot}$ are deterministic functions of $V_{<t}$ and public parameters, and both parties condition on the same published prefix whether or not a proposal was rejected, so they compute the same alphabet size at every step and index the symbols identically; a rejection changes Y_t and possibly $\widehat{C}_t$, but not K_t or the position of the symbol. Unmasking is a bijection of $\mathbb{Z}_{K_t}$, so $\widehat{C}_t \neq C_t$ if and only if $f_{\mathsf{Bucket},t}(Y_t) \neq A_t$, which by Lemma 1(iii) happens if and only if the proposal is rejected. Given $\mathscr{F}_{t-1}$ and A_t, the proposal is accepted with probability $\min\{1, K_t \pi_t(A_t)\}$; averaging over a uniform A_t gives $\frac{1}{K_t}\sum_a (1 - K_t \pi_t(a))_+ = \sum_a (\frac{1}{K_t} - \pi_t(a))_+ = \Delta_{K_t}(p_t)$, in agreement with Lemma 1(ii). □

D.4 Proof of Theorem 2

Theorem 2 (Soundness)**.** *If* Tag *is computationally secure, the scheme is sound (Definition 3) for every decoder. Concretely, if the receiver validates at most Q candidate messages ($Q = 1$ in the warm-up), there is a PPT distinguisher $\mathscr{A}_2$ against* Tag *such that* Extract *returns a message other than m or $\perp$ with probability at most $Q2^{-\lambda} + \mathrm{Adv}^{\mathsf{prf}}_{\mathsf{Tag},\mathscr{A}_2}(\kappa)$.*

Proof. Let Bad be the event that a wrong candidate passes validation, that is, Extract returns some $\widehat{m} \notin \{m, \perp\}$, which requires its decoded tag to equal $\mathsf{Tag}_{k_v}(\mathsf{sid}, \widehat{m})$. Let $\mathscr{A}_2$ be the PPT distinguisher that runs the protocol with its oracle in place of Tag_{k_v} and outputs 1 if Bad occurs. Replacing Tag_{k_v} by a uniformly random function R onto $\{0,1\}^\lambda$ changes $\mathbb{P}[\mathsf{Bad}]$ by at most $\mathrm{Adv}^{\mathsf{prf}}_{\mathsf{Tag},\mathscr{A}_2}$, so it suffices to bound $\mathbb{P}[\mathsf{Bad}]$ when R is used. In this case, embedding evaluates R only at (sid, m), so the transcript, and hence every candidate produced from it, is independent of the values of R at all other points. Validating a candidate $\widehat{m} \neq m$ compares its decoded tag with $R(\mathsf{sid}, \widehat{m})$, a fresh uniform value, so the comparison succeeds with probability $2^{-\lambda}$. At most Q distinct candidates are validated, and a union bound gives $\mathbb{P}[\mathsf{Bad}] \leq Q2^{-\lambda}$. □

D.5 Proof of Lemma 2

Lemma 2 (Labeled lifted exact cover)**.** *If the symbol a is uniform on $\mathbb{Z}_q$ and independent of the pool, the published block has distribution p_i. If the symbols of all positions are independent and uniform, and independent of the pools, the published blocks are distributed as an honest transcript.*

Proof. Fix the pool $X_0, \ldots, X_{L-1}$ and its label table, which fixes the bijection Assign. The symbol a is uniform and Algorithm 2 draws J uniformly, so the pair (a, J) is uniform over a set of size L, and Assign maps this set bijectively onto the L candidate indices. Hence the published index $I = \mathsf{Assign}(a, J)$ is uniform over the indices. This holds for every pool and label table, so I is independent of the pool. Now let the pool vary. Its blocks are independent samples from p_i, and I is uniform and independent of them, so the published block $X = X_I$ has distribution p_i. Duplicate blocks and unbalanced labels do not change this indices-level argument. The composition over positions follows from the chain rule. Condition on the public history before position i, which includes all earlier published blocks. Given this history, the pool at position i consists of independent samples from p_i, and the symbol is uniform and independent of the pool by assumption. The argument above then shows that the published block at position i has distribution p_i given the history. An honest transcript is generated by exactly these conditional distributions, so the joint distribution of all published blocks equals that of an honest transcript. □

The proof of Lemma 3 also uses the following fact about the labels that the reader obtains in the first phase.

Claim 1 (Labels under the identity serializer)**.** *With the identity serializer, the reader obtains the label $y = \mathsf{Hash}_q(\mathsf{sid}, i, X)$ of the published block X at every position i, and these labels are uniform on $\mathbb{Z}_q$, independent of the published blocks, and independent across positions.*

Proof. In the random oracle model, the value of the hash at the fresh input (sid, i, X) is uniform on $\mathbb{Z}_q$ for every fixed X, so y is uniform and independent of X. Note that y differs from the symbol a by the displacement of the carrier, and this displacement is correlated with a. The uniformity of y does not rely on the displacement, but only on y being the hash value at a uniformly selected block. Inputs at different positions and in different exchanges are distinct, so their labels are independent. □

D.6 Proof of Lemma 3

Lemma 3 (Joint pseudorandomness of the transmitted symbols)**.** *Let $\mathscr{A}$ be a PPT auditor that reads the completed transcript. Feeding the carrier independent uniform symbols in place of the coefficients of b, w, and z changes the probability that $\mathscr{A}$ outputs 1 by at most $\mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}(\kappa) + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}(\kappa)$, for PPT distinguishers $\mathscr{B}_1$ and $\mathscr{B}_2$ built from $\mathscr{A}$.*

Proof. We use a hybrid argument over three experiments, where the auditor reads the completed transcript.

- H_0: the real exchange of Algorithm 3.
- H_1: identical to H_0, except that the receiver feeds the carrier a uniform vector $u_b \sim \mathscr{U}(R_q^k)$ in place of $b = As_{\mathrm{R}} + e_{\mathrm{R}}$; the sender is unchanged and computes w and z from the labels $\widetilde{b}$ it reads.
- H_2: identical to H_1, except that the sender feeds the carrier a uniform pair $(u_w, u_z) \sim \mathscr{U}(R_q^k \times R_q)$ in place of (w,z); this is the modified experiment of the lemma.

Write $\mathbb{P}[\mathsf{H}_j = 1]$ for the probability that $\mathscr{A}$ outputs 1 in H_j.

$\mathsf{H}_0 \approx \mathsf{H}_1$. Let $\mathscr{B}_1$ be the distinguisher that receives (A,t) with $t \in R_q^k$, runs the exchange with A as the public matrix and the coefficients of t as the receiver's symbols, simulates the sender honestly, runs $\mathscr{A}$ on the resulting transcript, and outputs the output of $\mathscr{A}$. When $t = As_{\mathrm{R}} + e_{\mathrm{R}}$, $\mathscr{B}_1$ simulates H_0 exactly; when t is uniform, it simulates H_1 exactly. Hence

$$\left|\mathbb{P}[\mathsf{H}_0 = 1] - \mathbb{P}[\mathsf{H}_1 = 1]\right| = \mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}.$$

$\mathsf{H}_1 \approx \mathsf{H}_2$. In H_1, the symbols of the first phase are uniform, so by Lemma 2 the published blocks of the first phase are honest samples, and by Claim 1 the vector $\widetilde{b}$ of their labels is uniform on R_q^k and independent of the blocks and of A. Here $\widetilde{b}$ is the vector of hash values $\mathsf{Hash}_q(\mathsf{sid}, i, X_i)$ of the published blocks, and its uniformity comes from the hash; the carrier displacement $\widetilde{b} - u_b$ is correlated with u_b, but u_b enters H_1 only through the selection of the blocks. Let

$$M = \begin{pmatrix} A^T \\ \widetilde{b}^T \end{pmatrix} \in R_q^{(k+1)\times k}, \qquad e = (e_1, e_2) \in R_q^{k+1},$$

so that M is uniform and the sender's symbols in H_1 are

$$(w,z) = Ms_{\mathrm{S}} + e + (0, \iota(c^+)).$$

In H_2 the sender's symbols are (u_w, u_z), and $(u_w, u_z) - (0, \iota(c^+))$ is again uniform, so the two experiments differ exactly in whether $Ms_{\mathrm{S}} + e$ or a uniform element of R_q^{k+1} is added to $(0, \iota(c^+))$. Let $\mathscr{B}_2$ be the distinguisher that receives (M, t') with $t' \in R_q^{k+1}$ and proceeds as follows.

(a) It uses the first k rows of M as A^T and reads the last row as $\widetilde{b} \in R_q^k$.
(b) For each position i of the first phase, it samples the pool honestly, publishes a uniformly chosen pool member X_i, and computes $\mathsf{Hash}_q(\mathsf{sid}, i, X_i)$ as the coefficient of $\widetilde{b}$ that position i carries. By Lemma 2 and Claim 1, the published block and its label have exactly their joint distribution in H_1, and no earlier oracle query constrains the computed hash value.
(c) It runs the second and third phases of the carrier with the symbols $t' + (0, \iota(c^+))$, runs $\mathscr{A}$ on the completed transcript, and outputs the output of $\mathscr{A}$.

When $t' = Ms_{\mathrm{S}} + e$, $\mathscr{B}_2$ simulates H_1 exactly; when t' is uniform, it simulates H_2 exactly. Hence

$$\left|\mathbb{P}[\mathsf{H}_1 = 1] - \mathbb{P}[\mathsf{H}_2 = 1]\right| = \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}.$$

By the triangle inequality,

$$\left|\mathbb{P}[\mathsf{H}_0 = 1] - \mathbb{P}[\mathsf{H}_2 = 1]\right| \le \mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1} + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2},$$

and both distinguishers run $\mathscr{A}$ once plus one execution of the exchange, so they are PPT whenever $\mathscr{A}$ is. □

D.7 Proof of Theorem 3

Theorem 3 (Undetectability and key security)**.** *For every PPT passive auditor $\mathscr{A}$ with at most Q_{KDF} KDF queries, there are PPT $\mathscr{B}_1$ and $\mathscr{B}_2$ with*

$$\mathrm{Adv}^{\mathsf{cover}}_{\mathscr{A}}(\kappa) \le \mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}(\kappa) + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}(\kappa),$$
$$\mathrm{Adv}^{\mathsf{key}}_{\mathscr{A}}(\kappa) \le \mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1}(\kappa) + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}(\kappa) + \frac{Q_{\mathsf{KDF}}}{2^{\ell}}.$$

Proof. By Lemma 3, the auditor's view is indistinguishable, up to $\mathrm{Adv}^{\mathsf{mlwe}}_{k,\mathscr{B}_1} + \mathrm{Adv}^{\mathsf{mlwe}}_{k+1,\mathscr{B}_2}$, from the transcript of an exchange whose carrier is fed independent uniform symbols at every position. By Lemma 2, composed over positions and applied with slots indexed by $\mathbb{Z}_q$ regardless of the serializers, that transcript consists of honest samples, which proves the first bound of Theorem 3.

In the modified experiment, the transcript is independent of the pre-key K_0, so the sender's key $\mathsf{KDF}(K_0, V)$ is uniform unless the auditor queries the KDF at the point (K_0, V), which happens with probability at most $Q_{\mathsf{KDF}}/2^{\ell}$, and this proves the second bound. □

D.8 Proof of Theorem 4

Theorem 4 (Noise-resilient decoding)**.** *For arbitrary displacements $(\zeta_b, \zeta_w, \zeta_z)$, where ζ_w and ζ_z include the serializer reconstruction error, the receiver's residual is*

$$\omega = \iota(c^+) + (e_{\mathrm{R}} + \zeta_b)^T s_{\mathrm{S}} - s_{\mathrm{R}}^T(e_1 + \zeta_w) + e_2 + \zeta_z \pmod q,$$

and if $W = d_H(\mathsf{HD}(\omega)_{[N]}, c) \le \tau$, the decoder recovers K_0 and $K_{\mathrm{R}} = K_{\mathrm{S}}$.

Proof. Substitute $b = As_{\mathrm{R}} + e_{\mathrm{R}}$, $w = A^T s_{\mathrm{S}} + e_1$, and $z = \widetilde{b}^T s_{\mathrm{S}} + e_2 + \iota(c^+)$ into $\omega = \widetilde{z} - s_{\mathrm{R}}^T \widetilde{w}$. The two copies of $s_{\mathrm{R}}^T A^T s_{\mathrm{S}}$ cancel, which yields the residual stated in Theorem 4. The ECC corrects every error pattern of weight at most τ, so the recovered pre-key and the transcript agree at both parties, and so do their KDF outputs. □

D.9 Proof of Theorem 5

Theorem 5 (Empirical correctness)**.** *Let $\Omega_1, \ldots, \Omega_m \sim \mathscr{D}$ be independent complete sessions and $S < m$ the number of them with $W > \tau$. With probability at least $1 - \alpha$ over the draw of the m sessions,*

$$\delta_{\mathscr{D}} \le \mathbb{P}_{\mathscr{D}}[W > \tau] \le \mathrm{Beta}^{-1}(1 - \alpha; S + 1, m - S).$$

Proof. The indicators $F_i = \mathbf{1}\{W_i > \tau\}$ of the sessions are independent Bernoulli variables with mean $\mathbb{P}_{\mathscr{D}}[W > \tau]$, the bound is the exact one-sided Clopper–Pearson bound for their mean, and Corollary 5 transfers it to key disagreement. □