Unforgeable Watermarks for Language Models via Robust Signatures

Huijia Lin[a], Kameron Shahabi[a], and Min Jae Song[b]

[a]Paul G. Allen School of Computer Science & Engineering, University of Washington
[b]Data Science Institute, University of Chicago
{rachel,kshahabi}@cs.washington.edu, minjae.song@uchicago.edu

February 17, 2026

Abstract

Language models now routinely produce text that is difficult to distinguish from human writing, raising the need for robust tools to verify content provenance. Watermarking has emerged as a promising countermeasure, with existing work largely focused on model quality preservation and robust detection. However, current schemes provide limited protection against false attribution.

We strengthen the notion of soundness by introducing two novel guarantees: unforgeability and recoverability. Unforgeability prevents adversaries from crafting false positives, texts that are far from any output from the watermarked model but are nonetheless flagged as watermarked. Recoverability provides an additional layer of protection: whenever a watermark is detected, the detector identifies the source text from which the flagged content was derived. Together, these properties strengthen content ownership by linking content exclusively to its generating model, enabling secure attribution and fine-grained traceability.

We construct the first undetectable watermarking scheme that is robust, unforgeable, and recoverable with respect to substitutions (i.e., perturbations in Hamming metric). The key technical ingredient is a new cryptographic primitive called robust (or recoverable) digital signatures, which allow verification of messages that are close to signed ones, while preventing forgery of messages that are far from all previously signed messages. We show that any standard digital signature scheme can be boosted to a robust one using property-preserving hash functions (Boyle, LaVigne, and Vaikuntanathan, ITCS 2019).

Contents

1 Introduction **3**
- 1.1 Our results . . . 6
- 1.2 Related work . . . 12

2 Technical Overview **14**
- 2.1 Robust and recoverable signatures from PPH . . . 14
- 2.2 Watermarking using robust signatures and block steganography . . . 15
- 2.3 Discussion . . . 18

3 Watermarking definitions **20**
- 3.1 Watermarking . . . 21

4 Preliminaries **24**
- 4.1 Property-preserving hash . . . 26
- 4.2 Block steganography . . . 27

5 Robust and Recoverable Digital Signatures **33**
- 5.1 Difference-recovering hash family . . . 34
- 5.2 Robust signature from PPH . . . 36
- 5.3 Recoverable signatures from difference-recovering PPH . . . 37
- 5.4 Instantiating Hamming signatures . . . 38

6 Recoverable, Robust, and Unforgeable Watermarking **39**
- 6.1 Robust and unforgeable watermarking . . . 39
- 6.2 Generalizing verification to chains . . . 42
- 6.3 Recoverable watermarking . . . 46
- 6.4 Instantiating Hamming watermarks . . . 50

A Strong Difference Recovery and Homomorphic PPH **57**

B Sharp Sketches and Lower Bounds **58**

1 Introduction

The rapid rise of generative AI has made reliable content provenance—maintaining verifiable records of the origin and history of generated content—an urgent priority. The remarkable quality of content output by GenAI models, including large language models (LLMs) like GPT-4 [AAAA+23] and diffusion models such as Stable Diffusion [RBLE+22], increasingly blurs the line between authentic and synthetic content. Without reliable provenance, trust in digital ecosystems erodes, enabling large-scale misinformation, disputed ownership, and other serious risks [BBBC+23]. This paper focuses on two avenues of content provenance:

- *Tracing content to its source*: Attributing generated content to the model, model owner, or prompter responsible for generating it.
- *Tracking edits*: Recovering the modifications or edits made to generated content by computationally bounded parties.

A recent line of work [Aar22; KGWK+23; GZS24] has introduced *watermarking* as a tool for provenance. However, existing notions of watermarking typically target a narrow objective and offer guarantees that fall short of reliable source tracing. Moreover, prior approaches do not consider the challenge of tracking edits. In this work, we address these limitations in the context of language models.

Existing watermarking guarantees. A watermarking scheme for a language model is a protocol for embedding hidden, verifiable watermarks into generated content. It consists of three algorithms, $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$. The key-generation algorithm jointly samples watermarking and verification keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$. As we will discuss shortly, we may either consider a public or secret verification key. The watermarking algorithm $\mathsf{Wat}(\mathsf{sk}, \cdot)$ acts as the interface to the language model, receiving prompts $x \in \{0,1\}^*$ and producing watermarked responses $y \in \{0,1\}^*$. The verification algorithm $\mathsf{Ver}(\mathsf{vk}, \cdot)$ decides whether a candidate string $\zeta \in \{0,1\}^*$ is watermarked. Existing constructions [CG24; GM24; FGJM+25] satisfy the following security properties:

- *Robustness:* No efficient attacker with oracle access to $\mathsf{Wat}_{\mathsf{sk}}$[1] can produce text that is *close*, under a specified closeness predicate, to one of the responses $y \leftarrow \mathsf{Wat}_{\mathsf{sk}}(x)$ that they observe, but for which verification fails. In other words, it is hard to remove the watermark through mild modifications.
- *Soundness:* No attacker acting *independently* of the watermarking and verification keys $(\mathsf{sk}, \mathsf{vk})$ can generate text that is incorrectly flagged as watermarked.

Preserving model quality is also a central goal. The strongest notion, *undetectability* [CGZ24, Definition 9], requires that without the watermarking and verification keys, no efficient observer can distinguish the oracle $\mathsf{Wat}_{\mathsf{sk}}$ from an oracle that produces standard (unwatermarked) outputs from the underlying model. In this work, we adopt undetectability as a core requirement for watermarking schemes.

At a high level, robustness and soundness address complementary failures: robustness protects against adversaries attempting to trigger false negatives, while soundness defends against false positives. However, the threat model for soundness is weaker than that for robustness. Soundness adversaries must act *without knowledge* of the watermarking and verification keys. This restriction

[1] In the public verification key setting, the attacker also receives the verification key vk; in the secret key setting they instead have oracle access to $\mathsf{Ver}_{\mathsf{vk}}$.

is information-theoretic, implying that the attacker has no access to watermarked content (which depends on the keys) and cannot test verification decisions on candidate text. In contrast, robustness adversaries may adaptively query watermarked outputs and are provided access to the verification procedure [2].

Despite this gap, these definitions do well in capturing the original use cases of language watermarking: distinguishing AI-generated content from honest, human-written text. Prior work has focused on two settings: *public verification* [FGJM+25], where the verification key is released so that anyone can distinguish content, and *private verification* [CG24; GM24], where a model owner keeps their verification key secret and uses it to test their own model's content—for example, to avoid retraining their model on its own prior outputs while scraping the web for training data [SSZP+24]. The robustness adversary then receives appropriate access to the verification key—full access to vk in the public key setting and oracle access to $\mathsf{Ver}_{\mathsf{vk}}$ in the secret key setting—as well as oracle access to $\mathsf{Wat}_{\mathsf{sk}}$, reflecting the capabilities of a motivated attacker who is actively trying to remove the watermark. The soundness adversary, on the other hand, gets no access to the keys, which is justified under this traditional setup, as honest humans are unlikely to produce text correlated with randomly chosen watermarking and verification keys, and thus would rarely trigger false positives. Moreover, some applications can naturally tolerate a small number of false positives. For instance, in the example of filtering a model's own outputs from its training data, it may be acceptable if a few human written texts are mistakenly identified as watermarked and filtered out of the training data.

Tracing the source: Unforgeability. Unfortunately, existing soundness guarantees are insufficient for reliably tracing content back to its source. This is an important functionality for applications that require accountability, where it is crucial to identify parties responsible for generating harmful content, or important to assign credit to deserving parties that produced content.

One might hope that verifying text under a particular verification key, together with soundness, would serve as proof that the content originated from the owner of the corresponding watermarking key. However, enabling such fine-grained attribution motivates a new class of attacks. Adversaries may observe previously generated watermarked content, detection outcomes (in the secret key setting), or the verification key itself (in the public key setting) to produce content that differs significantly from any genuine model output, yet is still flagged as watermarked. We refer to such attacks as *forgeries*. For example, an adversary may want to forge a model owner's watermark onto harmful content, to frame the model owner for producing it. Soundness alone, which only protects against key-independent adversaries, provides no protection against these key-dependent forgeries.

To address this, we propose a strengthened notion of soundness, (robust) unforgeability, which allows for dependence on keys:

- *Unforgeability:* No efficient attacker with a watermarking oracle $\mathsf{Wat}_{\mathsf{sk}}$ can produce content that is *far* (under some closeness predicate) from all the responses $y \leftarrow \mathsf{Wat}_{\mathsf{sk}}(x)$ that they observe, yet is still flagged as watermarked.

The attacker is also given access to the verification key vk (in the public key setting) or to a verification oracle $\mathsf{Ver}_{\mathsf{vk}}$ (in the secret key setting). Figure 1 illustrates the distinction between soundness and unforgeability. With an unforgeable watermarking scheme, content that verifies under a particular verification key vk can be reliably attributed to the owner of the corresponding watermarking key sk, since it must *closely resemble* content generated using that key.

[2] Because soundness is defined under a weaker threat model, existing watermarking schemes typically achieve soundness against computationally unbounded adversaries, whereas robustness has been achieved either against computationally bounded [CG24; AACD+24] or unbounded [FGJM+25] adversaries.

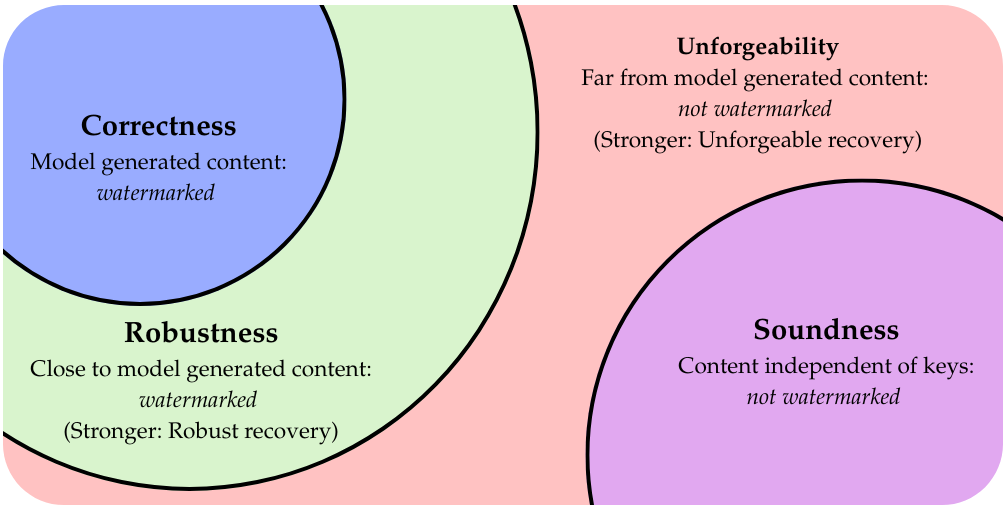


Figure 1: Illustration of watermarking guarantees. Correctness and soundness (blue and purple) are statistical guarantees, whereas robustness and unforgeability (green and red) can be computational. Consequently, the green and red regions may *contain* unwatermarked and watermarked content respectively, but no efficient adversary can find them.

Our notion of unforgeability is inspired by digital signatures but differs in that it only prevents forgery of content that is significantly *far* from the model's outputs, rather than content that is merely different. This design ensures that unforgeability complements robustness. A strict digital signature–style definition would conflict with robustness, since even mildly modified text that still verifies could be treated as a forgery. Prior work on watermark unforgeability [CG24; FGJM+25] illustrates this trade-off: these approaches leverage digital signatures but do not maintain robustness.

Tracking edits: Recoverability. Prior works have yet to consider tracking edits to generated text. We introduce a formal notion, (robust) recoverability, which allows doing so. Recoverable watermarks include an additional polynomial time algorithm Rec which outputs a list, with the property:

- *Robust recoverability:* If an efficient adversary produces text ζ that is *close* (under a closeness predicate) to some response $y \leftarrow \mathsf{Wat}_{\mathsf{sk}}(x)$, then the list $\mathsf{Rec}(\mathsf{vk}, \zeta)$ contains a substring of y whose length is nearly the length of ζ[3].

A key benefit of recoverability is that it reduces ambiguity in defining *closeness* using a boolean predicate $\Phi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. In an unforgeable watermarking scheme, adversarially generated text verifies only if it is close, according to Φ, to one of the adversary's observed model-generated responses. However, text may be close under *that* predicate yet still not warrant attribution to the watermarker, as it seems unlikely that any single predicate can capture diverse attribution decisions. Recoverability addresses this by enabling reconstruction of the original output, allowing the verifier to make a more informed judgement about whether to attribute content to the watermarker or to a user who modified that content.

Recoverability can be viewed as a strengthening of robustness, protecting against false negatives by ensuring that we *recover*, rather than merely verify, when text is close to a watermarked output. It does *not*, however, protect against false positives in the recovered list. Nonetheless,

[3]We allow a small loss in length between the recovered substring from y and the adversary's output ζ.

some applications of recoverability (e.g., aiding in attribution decisions) may require the guarantee that the recovered substrings genuinely originate from the source model. We formalize this with *unforgeable recoverability*: if an efficient adversary produces text ζ such that the recovery algorithm outputs a non-empty list ($\mathsf{Rec}(\mathsf{vk}, \zeta) \neq \perp$), then each recovered $\xi \in \mathsf{Rec}(\mathsf{vk}, \zeta)$ must be a substring of some response $y \leftarrow \mathsf{Wat}_{\mathsf{sk}}(x)$ that the adversary approximately copied.

Applications of unforgeability and recoverability. Unforgeability enables reliably tracing content back to a specific *watermarking key*. This supports the following applications:

- *Strengthened claims of ownership and accountability.* Unforgeable watermarking guarantees that content attributed to a particular watermarker (e.g., a model owner or a prompter) genuinely originates from them. This enables reliable authorship claims (e.g., for AI-generated art) and supports accountability (e.g., determining whether harmful content was actually produced by a certain watermarker).
- *Improved soundness for honest users.* A user may interact with a watermarked model honestly—for instance, by observing and substantially modifying watermarked outputs to avoid direct copying. Soundness offers no protection against false positives in such cases, since the resulting text may still be correlated with the watermarking key. Moreover, as language models and watermarked outputs become increasingly widespread, even fully human-written text may inadvertently be correlated with watermarking keys [YLBS+24], undermining the independence assumption that soundness relies on. Our threat model protects such honest users from false positives.
- *Tracking memorization or distillation.* Consider a scenario in which a new language model, trained on content scraped from the internet, may have memorized outputs from a different, watermarked model. Unforgeability guarantees that if the new language model's output verifies as watermarked under that watermarking key, then it is genuinely close to an output produced by the watermarked model. This provides strong evidence for claims of model distillation (as alleged in DeepSeek's use of GPT-4).

Recoverability, in turn, allows for tracking the modifications made to generated content produced by a certain watermarking key. This allows:

- *Finer-grained attribution.* For example, in the public verification key setting, an instructor may recover AI-generated text that a student approximately copied, in order to determine whether that student used AI in a permissible manner. Unforgeable recoverability ensures that the student truly observed edit the recovered AI-generated text.
- *Tracing edit histories.* A verifier can reconstruct content that has been modified or built upon by multiple watermarkers by following the recovery chain using their respective verification keys.

1.1 Our results

Our contributions fall under two categories: conceptual and technical. Conceptually, we introduce new public and secret key notions of unforgeability and recoverability for watermarking schemes (Section 3.1). Towards attaining these properties, we define new cryptographic tools—robust and recoverable digital signatures (Section 5)—which may be of independent interest. We show these can be constructed from property-preserving hashes (PPHs) [BLV19], a primitive that, although

previously studied, has largely been treated as a standalone tool. Our work demonstrates a novel use of PPH as a building block in cryptographic constructions, uncovering an unexplored connection to AI content provenance.

On the technical side, we develop a general framework for constructing robust and recoverable digital signatures, as well as recoverable, robust, and unforgeable watermarking schemes for language model. Our framework accomodates any general predicate defining closeness. We demonstrate its applicability by obtaining both public and secret key watermarking schemes for the Hamming predicate (Section 6.4). Along the way, we identify and formalize a useful property of PPHs, which we call *difference recovery* (Section 5.1), and show that every PPH for the Hamming predicate must satisfy this property.

Defining robust, unforgeable, and recoverable watermarking. In Section 3.1, we present our new notions of security for watermarking schemes. We model adversaries as probabilistic polynomial-time (PPT) algorithms that may interact with oracles. For an adversary $\mathcal{A}$ with oracle access to M, the *transcript* of their interaction, denoted $\mathsf{Trans}(\mathcal{A}^{\mathsf{M}})$, is the set of all input–output pairs observed by $\mathcal{A}$. For example, if M is a watermarking oracle $\mathsf{Wat}_{\mathsf{sk}}$, the transcript is the set of all the prompt-response pairs (x, y) such that $\mathcal{A}$ queried $\mathsf{Wat}_{\mathsf{sk}}(x)$ and received response y.

A robust and unforgeable watermarking scheme ensures that adversarially generated text verifies if and only if it is close to one of the adversary's observed responses. This can be decomposed into two conditions: the "if" direction corresponds to robustness, while the "only if" direction corresponds to unforgeability. The meaning of closeness is determined by a two-input predicate Φ that outputs 1 iff its inputs are deemed close.

Definition 1.1 (Robust and unforgeable watermarking, informal version of Definitions 3.4 and 3.5)**.** *For any closeness predicate Φ, a watermarking scheme $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$ for a language model is public key Φ-robust and unforgeable if for any PPT adversary $\mathcal{A}$,*

$$\Big(\mathsf{Ver}(\mathsf{vk}, \zeta) = 1\Big) \quad \Longleftrightarrow \quad \Big(\exists (x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}) \text{ such that } \Phi(\zeta, y) = 1\Big)\,,$$

with probability at least $1 - \mathrm{negl}(\lambda)$ over $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\zeta \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$.

Secret key robustness and unforgeability are the same, except $\mathcal{A}$ gets oracle access to $\mathsf{Ver}_{\mathsf{vk}}$, rather than full access to vk.

Note that the adversary's output $\zeta \in \{0,1\}^*$ and watermarked outputs $y \in \{0,1\}^*$ may have different lengths. With this in mind, one can think of $\Phi(\zeta, y) = 1$ semantically as

"ζ is approximately contained in y,"

meaning that there exists a substring of y to which ζ is similar to. In other words, $\Phi(\zeta, y) = 1$ indicates that the adversary approximately copied a portion of the response y.

We also define *recoverable* watermarking. Robust recovery states that a verifier can recover an original substring from *every* watermarked response that an adversary approximately copies from.

Definition 1.2 (Robust recovery, informal version of Definition 3.6)**.** *For any closeness predicate Φ, a* recoverable *watermarking scheme $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Rec})$ for a language model is public key Φ-robust if for any PPT adversary $\mathcal{A}$,*

$$\begin{aligned}&\forall (x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}) \text{ such that } \Phi(\zeta, y) = 1,\\ &\mathsf{Rec}(\mathsf{vk}, \zeta) \text{ contains a substring of } y \text{ that is the same size as } \zeta, \text{ up to some loss,}\end{aligned}$$

with probability at least $1 - \mathrm{negl}(\lambda)$ over $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\zeta \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$.

We require that any recovered substring be sufficiently long relative to the adversary's output, meaning that $\mathsf{Rec}(\mathsf{vk}, \zeta)$ contains a substring of size at least $(|\zeta| - \text{loss})$ for a prescribed loss parameter.

Robust recoverability does not protect against false positives in the recovered list; this restriction is enforced separately by the following unforgeability requirement, which ensures that every recovered string corresponds to a genuine model output that was approximately copied by the adversary.

Definition 1.3 (Unforgeable recovery, informal version of Definition 3.7)**.** *For any closeness predicate Φ, a* recoverable *watermarking scheme* $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Rec})$ *for a language model is public key Φ-unforgeable if for any PPT adversary $\mathcal{A}$,*

$$\begin{aligned} &\textit{for every recovery } \xi \in \mathsf{Rec}(\mathsf{vk}, \zeta) \\ &\exists (x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}}) \textit{ such that } \xi \textit{ is a substring of } y \textit{ and } \Phi(\zeta, y) = 1, \end{aligned}$$

with probability at least $1 - \mathrm{negl}(\lambda)$ *over* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $\zeta \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$.

Recoverability can thus be viewed as a *witness* variant of robustness: rather than merely deciding whether a given text ζ is watermarked, Rec outputs a certificate justifying that decision—namely a list of model-generated texts whose similarity to ζ can be verified. Any recoverable watermarking scheme that satisfies robustness and unforgeability immediately yields a standard robust and unforgeable watermarking scheme: one can simply verify content whenever the recovery algorithm recovers any text, $\mathsf{Ver}(\mathsf{vk}, \zeta) := (\mathsf{Rec}(\mathsf{vk}, \zeta) \neq \bot)$.

Realizing our definitions. Our main technical result is a general framework for constructing watermarking schemes that are recoverable, robust, and unforgeable. This framework relies on two building blocks. The first is *robust steganography*, an abstraction which we observe is implicitly achieved by existing watermarking schemes. The second is *robust and recoverable digital signatures*, a new notion that we introduce and construct.

Robust block steganography. Robust block steganography schemes (Definition 4.11) are closely related to watermarking schemes, but additionally allow embedding messages into generated content while preserving output quality. The term *block* refers to their fixed-size interface: for a given security parameter, the message size k and output size n are fixed, and the scheme embeds messages $m \in \{0,1\}^k$ into *blocks* $y \in \{0,1\}^n$.

More formally, for message and block sizes $k, n : \mathbb{N} \to \mathbb{N}$, a block steganography scheme for a language model consists of algorithms $(\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$. The key-generation algorithm samples secret and decoding keys jointly as $(\mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$. The embedding algorithm $\mathsf{Embed}(\mathsf{sk}, \pi, m) \in \{0,1\}^{n(\lambda)}$ takes a context $\pi \in \{0,1\}^*$ and a message $m \in \{0,1\}^{k(\lambda)}$ and outputs a block $y \in \{0,1\}^{n(\lambda)}$. The decoding algorithm $\mathsf{Dec}(\mathsf{dk}, \zeta) \in \{0,1\}^{k(\lambda)} \cup \{\bot\}$ outputs either a decoded message or a failure symbol.

Robustness with respect to a closeness predicate Φ requires that if a block $\zeta \in \{0,1\}^{n(\lambda)}$ (chosen by an efficient adversary) is close to a valid embedding y of a message m, then ζ will decode to that message,

$$\Big(\exists ((\pi, m), y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Embed}_{\mathsf{sk}}}) \text{ such that } \Phi(\zeta, y) = 1\Big) \implies \Big(\mathsf{Dec}(\mathsf{dk}, \zeta) = m\Big),$$

Block steganography should also preserve the quality of generated text. We enforce this by requiring they satisfy undetectability [CGZ24] with respect to the underlying language model[4].

[4]Informally, this ensures that no efficient adversary, without the watermarking and verification keys, can distinguish an oracle producing embedded outputs $\mathsf{Embed}_{\mathsf{sk}}$ from one producing unembedded outputs according to the underlying language model.

We emphasize that robustness and undetectability are the only properties we require. In particular, block steganography schemes need not prevent false positives nor support recovery, and therefore do not provide unforgeability or recoverability. This matches existing constructions [CG24; AACD+24; FGJM+25], which achieve neither[5]. Nonetheless, we show that block steganography can be generically composed with robust and recoverable digital signatures to obtain watermarking schemes that are simultaneously robust, recoverable, and unforgeable.

Robust signatures. We introduce new notions of robust and recoverable signatures (Section 5). A robust signature scheme verifies on an adversarially chosen message-signature pair (ζ, σ) if and only if ζ is *close* to a message y that was previously signed with σ. A signature scheme is further said to be *recoverable* if, from any adversarially chosen message-signature pair (ζ, σ), the recovery algorithm recovers a message y if and only if (1) ζ is close to y and (2) y is a message that was previously signed with σ.

Definition 1.4 (Robust/Recoverable signatures, informal version of Definitions 5.1 and 5.2)**.** *For any closeness predicate Φ, a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is Φ-robust if for any PPT adversary $\mathcal{A}$,*

$$\Big(\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma) = 1\Big) \iff \Big(\exists y \text{ such that } (y, \sigma) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}) \text{ and } \Phi(\zeta, y) = 1\Big) .$$

A digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Rec})$ is further said to be Φ-recoverable if

$$\Big(\mathsf{Rec}(\mathsf{pk}, \zeta, \sigma) \text{ outputs } y\Big) \iff \Big((y, \sigma) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}) \text{ and } \Phi(\zeta, y) = 1\Big) .$$

Here, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $(\zeta, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk})$,

We refer to the reverse direction of the above statements——namely, that (ζ, σ) verifies (or recovers) only if ζ is close to a message y signed with σ——as *strong* unforgeability. Strongly unforgeable robust signatures generalize standard strong UF-CMA digital signatures [BS07], which are recovered as the special case where Φ is the equality predicate. The term *strong* reflects the fact that this notion guarantees not only that signatures cannot be forged on unsigned messages, but also that no *new* signatures can be produced even for messages that were previously signed under a different signature.

Informal guarantees of our framework. Using these two tools, we design a general approach for constructing robust and unforgeable watermarking schemes. To understand its guarantees, it will be helpful to think of a string $y \in \{0,1\}^*$ as being broken into consecutive, non-overlapping n-bit *blocks* $(y_1, \ldots, y_{\lfloor |y|/n \rfloor})$, where y_t denotes the t-th contiguous, non-overlapping n-bit substring of y. We make this notation precise in Section 3.

At a high level, for a security parameter λ, our construction produces watermarked text $y \leftarrow \mathsf{Wat}_{\mathsf{sk}}(x)$ by repeatedly generating blocks y_t of size $n = n(\lambda)$ using a block steganography scheme. Each block embeds the robust signature σ_{t-1} of the preceding block y_{t-1}. Verification on candidate text $\zeta = (\zeta_1, \ldots, \zeta_{\lfloor |\zeta|/n \rfloor})$ proceeds by attempting to decode a signature from each block ζ_t and verifying it against the preceding block ζ_{t-1}[6]. Intuitively, this means that ζ will verify if and only if its blocks are close to a *sequence* of back-to-back blocks from a watermarked response y.

[5] These constructions do satisfy soundness, but our results hold even when using an unsound block steganography scheme.

[6] Section 2.2 contains a more detailed overview.

We capture this intuition using the *every-block-close* lifting of a predicate Φ, $\mathsf{EBC}[\Phi, n]$. This predicate deems that "ζ is approximately contained in y" if it is blockwise Φ-close to a consecutive sequence of blocks in y,

$$\mathsf{EBC}[\Phi, n](\zeta, y) : \begin{pmatrix} \text{Parse } \zeta \text{ and } y \text{ into } n\text{-bit blocks } (\zeta_1, \ldots, \zeta_{\lfloor |\zeta|/n \rfloor}) \text{ and } (y_1, \ldots, y_{\lfloor |y|/n \rfloor}). \\ \text{Output 1 iff } \exists t \in \mathbb{N} \text{ such that } \forall j \in [\lfloor |\zeta|/n \rfloor],\ \Phi(\zeta_j, y_{t+j-1}) = 1. \end{pmatrix}.$$

Figure 2 captures the predicate in action.

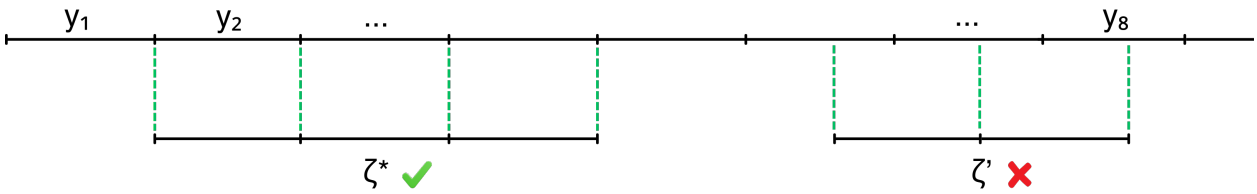


Figure 2: Illustration of $\mathsf{EBC}[\Phi, n]$-closeness. Green dotted lines indicate Φ-closeness. The string ζ^* is $\mathsf{EBC}[\Phi, n]$-close to y because its blocks are Φ-close to three consecutive blocks of y, whereas ζ' may be far: although ζ' is close to a substring of y of length $2n$, this substring is not aligned with two *blocks* of y.

By combining a Φ-robust block watermarking scheme, and a Φ-robust and recoverable digital signature scheme, we obtain a watermarking scheme that is recoverable, robust, and unforgeable with respect to the every-block-close lifting of Φ. Whether the watermark is public or secret key depends on the underlying block steganography scheme.

Theorem 1.5 (General watermarking framework, informal version of Theorem 6.1)**.** *Let Q be any language model, Φ be any closeness predicate, and $n : \mathbb{N} \to \mathbb{N}$ be any polynomial block size. Assume there exists both*

- *A (secret or public key) steganography scheme for Q that is robust to Φ and has block size n.*
- *A digital signature scheme that is robust to Φ.*

Then there exists a watermarking scheme for Q that is (secret or public key) robust and unforgeable[7] *with respect to the closeness predicate $\mathsf{EBC}[\Phi, n]$. If the digital signature scheme is additionally recoverable, then there exists a recoverable watermarking scheme for the same predicate.*

Remark 1.6 (Subtlety between robustness and unforgeability)**.** *While the informal theorem suggests that an input ζ verifies if and only if it is $\mathsf{EBC}[\Phi, n]$-close to a watermarked response, there is a subtle distinction in the formal guarantees:*

- *For* robustness*, verification succeeds if the* entire *string $(\zeta_1, \ldots, \zeta_{\lfloor |\zeta|/n \rfloor})$ is $\mathsf{EBC}[\Phi, n]$-close to a watermarked response y.*
- *For* unforgeability*, verification succeeds only if the substring consisting of all but the last block, $(\zeta_1, \ldots, \zeta_{\lfloor |\zeta|/n \rfloor - 1})$, is $\mathsf{EBC}[\Phi, n]$-close to a watermarked response y.*

This gap is made explicit in our full Theorems (Theorems 6.1 and 6.2).

[7] Against adversaries outputting sufficiently long strings ζ.

Instantiating the framework. We instantiate our framework to obtain a Hamming robust and unforgeable watermarking scheme. Prior work has already constructed both secret and public key robust block steganography schemes for any language model with sufficiently high entropy, when the predicate Φ is the Hamming predicate, which indicates if its inputs are close in Hamming distance. In particular, assuming the existence of ideal pseudorandom codes that are robust to a constant fraction of Hamming errors—shown by Alrabiah et al. [AACD+24] to follow from the subexponential hardness of exact learning parity with noise [JKPT12]—the watermarking construction of Christ and Gunn [CG24] yields a *secret key* steganography scheme that is robust to a constant fraction of Hamming errors (Theorem 4.14). Separately, in the random oracle model, Fairoze et al. [FGJM+25] construct a watermarking scheme for high-entropy language models, which can be viewed as a *public-key* steganography scheme that is robust to a subconstant fraction of Hamming errors (with a caveat, see the predicate Φ_δ defined in Theorem 4.16).

Finally, in this work we show that robust and recoverable digital signatures for a constant fraction of Hamming errors can be constructed under standard assumptions, namely the existence of collision-resistant hash functions and strong UF-CMA digital signatures (Corollary 5.11).

Using these results, we obtain both secret and public key recoverable, robust, and unforgeable watermarking schemes with respect to the every-block-close lifting of the Hamming predicate (Corollaries 6.6 and 6.7).

Constructing robust and recoverable signatures. In Section 5.2, we construct robust and recoverable digital signatures with respect to a constant fraction of Hamming errors. We do so by instantiating the following general framework, which leverages property-preserving hashing, introduced by Boyle et al. [BLV19].

Property-preserving hashing For any predicate Φ and hash size $k : \mathbb{N} \to \mathbb{N}$, a Φ-property preserving hash family (PPH) is a hashing scheme $(\mathsf{Sample}, \mathsf{Eval})$, that allows computing the predicate value of two inputs using only their hashes of length $k(\lambda)$. That is,

$$\mathsf{Eval}(h, h(y), h(y')) = \Phi(y, y')$$

with probability at least $1 - \mathrm{negl}(\lambda)$ over $h \leftarrow \mathsf{Sample}(1^\lambda)$ and $(y, y') \leftarrow \mathcal{A}(h)$. Importantly, Eval is adaptively secure.

We also observe that, although not explicitly stated, current PPH constructions for the Hamming predicate Ham_δ [FLS22; HLTW22] also allow recovering the bitwise difference $\Delta = x \oplus x'$ from a hash $h(x)$ and an input x' such that $\Phi(x, x') = 1$[8]. We formalize this as *difference recovery* and show that this is no coincidence: every PPH for the Hamming predicate must satisfy it (Proposition 5.5).

Definition 1.7 (Difference recovery, informal version of Definition 5.4)**.** *A PPH* $(\mathsf{Sample}, \mathsf{Eval})$ *for a predicate* Φ *is said to satisfy* difference recovery *if there exists a PPT recovery algorithm* Rec *such that*

$$\Big(\Phi(x, x') = 1\Big) \quad \Longrightarrow \quad \Big(\mathsf{Rec}(h, h(x), x') = x' \oplus x\Big) ,$$

holds with probability at least $1 - \mathrm{negl}(\lambda)$ *over* $h \leftarrow \mathsf{Sample}(1^\lambda)$ *and* $(x, x') \leftarrow \mathcal{A}(h)$.

Difference recovery is an unexplored aspect of PPHs, and our work illustrates its utility through a compelling application: recoverable watermarking.

[8] In fact, these constructions satisfy a stronger property, allowing recovery of the bitwise difference from just the hashes $h(x)$ and $h(x')$. However, we show that the weaker property, where the recovery algorithm gets x' in the clear, suffices for watermarking.

General framework for robust signatures. We show that any standard strongly unforgeable digital signature [BS07] can be supplemented with a Φ property preserving hash (with difference recovery) in order to build a Φ robust (recoverable) digital signature.

Theorem 1.8 (General robust signature framework, informal statement of Theorem 5.8)**.** *Let Φ be any closeness predicate and let $k, \ell : \mathbb{N} \to \mathbb{N}$ be any polynomials. Assume there exists*

- *A Φ-PPH with hash size k.*
- *A strongly unforgeable digital signature scheme with signature size ℓ.*

Then there exists a Φ-robust digital signature scheme with signature size $k + \ell$. Moreover, if the PPH is difference recovering, then the signature scheme is Φ-recoverable.

Instantiating this framework with the difference-recovering Hamming PPH of Holmgren et al. [HLTW22], which relies on a special homomorphic hash function, yields a signature scheme that is robust and recoverable against a constant fraction of Hamming errors, with signature size essentially a constant fraction of the message length (Corollary 5.11). For our purposes, however, we observe that it suffices to use a standard collision resistance hash function rather than the homomorphic one, giving us a Hamming recoverable digital signature scheme from CRHFs and strong UF-CMA signatures.

1.2 Related work

Watermarking schemes for generative models. Watermarking text has a history that predates the current era of large language models, with early work [TTD05; ARCH+01; ARHK+02] focused on embedding signals into *fixed* objects in a way that is hard to remove, while preserving their perceived quality. More relevant to our setting, which views language models as *randomized sampling algorithms*, is the recent line of work [Aar22; KGWK+23; KGWS+23; KTHL23; CGZ24; ZALW23] that develops watermarking techniques from this perspective, typically by embedding correlations with a watermarking key at the token level during the sampling process.

While many existing methods are distortion-free at the level of a single response [Aar22; KTHL23], they may introduce noticeable distortions when the watermarked model is queried repeatedly—across multiple rounds of interaction with arbitrary seed prompts. The strongest notion of model quality preservation that guards against such cumulative distortions, known as watermark *undetectability*, was introduced by Christ et al. [CGZ24].

Among undetectable watermarking schemes, only a select few guarantee unforgeability [FGJM+25; CG24]. These methods embed digital signatures into text in a computationally undetectable manner, thereby achieving both undetectability and unforgeability. However, these watermarks lack robustness. Our robust digital signatures can be integrated into either of these schemes. Recently, undetectable watermarking schemes for *image* models have been proposed by Gunn et al. [GZS24], representing a notable step toward achieving undetectability in generative models beyond discrete autoregressive sampling. For a broader overview of the field, we refer the reader to the surveys [LPLL+24; ZGCF+24].

Steganography. Steganography, the practice of hiding messages within content drawn from some distribution (referred to as a channel), has a long history of scientific study. Early work by Cachin [Cac98] considered a noiseless setting, where a message only needed to be hidden within unaltered content sampled from the channel. This study was motivated by Simmons' Prisoners' Problem [Sim84], in which two prisoners in separate cells wish to communicate hidden messages by embedding

them into letters visible to the warden. Hopper et al. [HLV02] further studied this problem and introduced robust steganography. However, in these initial works, the decoder was often assumed to know the distribution from which the content was drawn.

In contrast, hiding messages in AI-generated content presents additional challenges. The decoder, who sees content later down the line, does not know the prompt used to generate it and therefore cannot infer the distribution it came from. Nonetheless, a recent line of work has shown that (robust) steganography is still possible in this more demanding setting. Zamir [Zam24] demonstrated that, even without knowledge of the content distribution, messages can be hidden in unaltered content. Follow-up works [CG24; FGJM+25] extend this result to the noisy setting, constructing schemes in which the message can be decoded even in the presence of adversarial errors.

Pseudorandom error-correcting codes. Pseudorandom codes (PRC), introduced by Christ and Gunn [CG24], are error-correcting codes whose codewords are computationally indistinguishable from uniformly random strings to any observer lacking the decoding key. As objects that satisfy both pseudorandomness and error correctability, PRCs provide a natural foundation for constructing watermarking schemes that achieve both undetectability and robustness [CG24; GZS24; CHS24; GM24].

PRCs with varying robustness properties have been constructed under standard cryptographic assumptions, such as the subexponential hardness of learning parity with noise (LPN) [CG24; GG24]. In particular, Golowich and Moitra [GM24] construct PRCs that are robust to a constant fraction of edits, including substitutions, insertions, and deletions, at the cost of requiring a larger alphabet. This yields watermarking schemes that are robust under edit distance.

However, the notion of robustness satisfied by previous PRC constructions is *weak* in a different sense. The decoder's robustness guarantee holds only against corruptions that are generated *independently* of the PRC keys. In particular, this requirement assumes that the corruption adversary has not observed any prior codewords. This lack of *adaptive* robustness was highlighted by Cohen et al. [CHS24], who ask whether PRC messages remain robustly decodable even after the perturbation adversary (or error channel) has observed multiple codewords. Ideal PRCs, introduced by Alrabiah et al. [AACD+24], address this limitation by constructing PRCs that remain robust even against computationally bounded adversaries with black-box access to both encoding and decoding oracles. In this work, we use them to instantiate secret key robust block steganography schemes.

Property-preserving hash functions. We build our watermarking scheme on top of property-preserving hash functions (PPH), introduced by Boyle et al. [BLV19]. Several PPH constructions have been proposed for the exact Hamming predicate under standard cryptographic assumptions [FLS22; HLTW22; Zha25]. Particularly relevant to our work is the elegant construction by Holmgren et al. [HLTW22], which achieves nearly optimal hash size. Their key idea of combining syndrome list-decoding of linear error-correcting codes with collision-resistant hash functions has prior roots in the study of error-correcting codes against computationally bounded noise [MPSW10; DORS08].

Recent work by Bogdanov et al. [BRVV25] proposed and examined a new computational assumption that implies a PPH for the Gap-ℓ_2 predicate over Euclidean space. This enables evaluation of whether the ℓ_2 distance between a pair of real vectors is small or large based solely on their compressed digests. As such, these PPHs serve as *adversarially* robust locality-sensitive hash functions for the Euclidean distance: compressive hash functions for which it is computationally hard to find input pairs whose true pairwise distances differ substantially from those inferred from their hashes.

PPHs are also related to the cryptographic notion of *secure sketches* [DORS08], which are practically relevant in biometric applications where the secret key (e.g., one's biometric data) is inherently noisy and varies slightly with each use. Secure sketches enable recovery of a value x given a *sketch* $g(x)$ and a nearby input x'. This can be viewed as a relaxation of PPH, as it assumes access to one noisy input in the clear.

2 Technical Overview

We highlight the techniques we use to construct robust signatures (Section 2.1) and robust, unforgeable, and recoverable watermarks (Section 2.2).

2.1 Robust and recoverable signatures from PPH

Our construction produces a Φ-recoverable signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Rec})$ from a Φ-difference-recovering PPH, $\mathsf{PPH}[\mathsf{Sample}, \mathsf{Eval}, \mathsf{Rec}]$, and a strongly unforgeable digital signature scheme, $\mathsf{DSS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ [BS07]. An analogous construction yields a robust (but not recoverable) digital signature from a standard PPH without difference recovery.

To sign a message $y \in \{0,1\}^n$, we compute its property-preserving hash $h(y)$, then generate a signature of that hash $\tau \leftarrow \mathsf{DSS.Sign}(\mathsf{DSS.sk}, h(y))$. The robust signature is defined as $\sigma = (h(y), \tau)$. The public and secret keys are given by $\mathsf{pk} = (h, \mathsf{DSS.pk})$ and $\mathsf{sk} = \mathsf{DSS.sk}$.

To verify this signature $\sigma = (h(y), \tau)$ against a candidate message $\zeta \in \{0,1\}^n$, we compute

$$\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma) = \mathsf{DSS.Ver}(\mathsf{DSS.pk}, h(y), \tau) \wedge \mathsf{PPH.Eval}(h, h(y), h(\zeta)) \ .$$

We formally describe our construction in Figure 4. We give the intuition behind our security proofs, which can be found in Section 5.2.

Robustness. Let $\sigma = (z, \tau)$ be the robust signature of some message $y \in \{0,1\}^n$. If an adversarially chosen message ζ is Φ-close to y, then with overwhelming probability $\mathsf{PPH.Eval}(h, z, h(\zeta)) = 1$, since $z = h(y)$, from PPH correctness, and $\mathsf{DSS.Ver}(z, \tau) = 1$ from DSS correctness.

Unforgeability. Suppose an adversary $\mathcal{A}$ forges a pair (ζ, σ') with inverse-polynomial probability, meaning $\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma') = 1$, but ζ is Φ-far from all messages $y \in \{0,1\}^n$ associated with the signature σ'. We consider two cases and show that each leads to a contradiction.

1. The forged σ' is not one of the signatures observed by $\mathcal{A}$ through its queries to the robust signature oracle $\mathsf{Sign}_{\mathsf{sk}}$. Notice that responses from $\mathsf{Sign}_{\mathsf{sk}}$ are precisely the message-signature pairs $(z, \mathsf{DSS.Sign}(\mathsf{DSS.sk}, z))$ that $\mathcal{A}$ observes from the underlying standard strongly unforgeable digital signature DSS. If $\mathcal{A}$ outputs $\sigma' = (z', \tau')$ that verifies despite not being one of these pairs, then (z', σ') breaks the strong unforgeability of DSS.

2. The forged signature $\sigma' = (h(y), \tau)$ was observed by $\mathcal{A}$, meaning it is the signature of some message y that is Φ-far from the candidate message ζ. In this case, $\mathsf{PPH.Eval}(h, h(y), h(\zeta))$ detects, with overwhelming probability, that y and ζ are Φ-far. If it did not, the pair (y, ζ) would violate the security of the PPH. Consequently, the verifier rejects, $\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma') = 0$, except with negligible probability, leading to a contradiction.

Recoverability. For recoverability, we first (robustly) check $\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma')$ to ensure σ' is a signature of some y close to ζ. If the signature verifies, then output

$$\mathsf{Rec}(\mathsf{pk}, \zeta, \sigma') = \mathsf{PPH.Rec}(h, h(y), \zeta) \oplus \zeta \ .$$

Otherwise, $\mathsf{Rec}(\mathsf{pk}, \zeta, \sigma') = \perp$. Notice that when an adversarially chosen ζ is Φ-close to a previously signed message $y \in \{0,1\}^n$, difference recovery of PPH ensures that $\mathsf{PPH.Rec}(h, h(y), \zeta)$ outputs the bitwise difference $y \oplus \zeta$ with overwhelming probability, so $\mathsf{Rec}(\mathsf{pk}, \zeta, \sigma')$ correctly outputs y.

2.2 Watermarking using robust signatures and block steganography

Language models and blocks. Following prior work [GZS24; CG24; FGJM+25], we model a language model $Q : \{0,1\}^* \to [0,1]$ as a mapping that takes in a token sequence and outputs the probability that the next token is 1[9]. Given a prompt $x \in \{0,1\}^*$, we obtain an n-bit *response* $u = (u[1], u[2], \ldots, u[n]) \leftarrow \overline{Q}_n(x)$ by autoregressive sampling initialized with x.

Throughout this section, we refer to the n-bit *blocks* $(y_1, \ldots, y_{\lfloor |y|/n \rfloor})$ of a string $y \in \{0,1\}^*$. Informally, y_t denotes the t-th non-overlapping, n-bit substring of y.

A first attempt at unforgeable watermarking. A natural approach to constructing unforgeable watermarks is to sign text as it is generated using a standard digital signature scheme and then embed that signature into subsequently generated text via a block steganography scheme. Later, the signature can be decoded and verified along with the text. This is the approach taken by existing constructions that consider unforgeability [CG24; FGJM+25], but it comes at the cost of robustness.

Specifically, consider a language model Q. Prior work [CG24; AACD+24; FGJM+25] demonstrates that block steganography schemes can be constructed for any model with sufficiently high entropy. For the purposes of this overview, we therefore assume that Q has high entropy. We will attempt to watermark Q using the approach outlined above.

Let λ be the security parameter and $n = \mathrm{poly}(\lambda)$ be a block size. Let $\mathsf{Steg}[\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec}]$ be a Φ-robust block steganography scheme for Q with block size n. Steg can be either public or secret key. For the purposes of this overview, we will assume it's public key. Let $\mathsf{DSS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ be a *standard* digital signature scheme with message size n.

To generate verification and watermarking keys $(\mathsf{vk}, \mathsf{sk})$, sample

$$\mathsf{DSS.pk}, \mathsf{DSS.sk}) \leftarrow \mathsf{DSS.Gen}(1^\lambda), \quad (\mathsf{Steg.dk}, \mathsf{Steg.sk}) \leftarrow \mathsf{Steg.Gen}(1^\lambda)$$

and let $\mathsf{vk} = (\mathsf{DSS.pk}, \mathsf{Steg.dk})$, $\mathsf{sk} = (\mathsf{DSS.sk}, \mathsf{Steg.sk})$. We can generate $2n$ bits of watermarked text as follows. On input prompt $x \in \{0,1\}^*$, the watermarking algorithm $\mathsf{Wat}(\mathsf{sk}, x)$ first samples n bits normally from the underlying language model, $y_1 \leftarrow \overline{Q}_n(x)$. Next, it signs this block and embeds the signature into the following block[10]:

$$\sigma \leftarrow \mathsf{DSS.Sign}(\mathsf{DSS.sk}, y_1), \quad y_2 \leftarrow \mathsf{Steg.Embed}(\mathsf{Steg.sk}, \pi = (x, y_1), \sigma) \ ,$$

and outputs the concatenation (y_1, y_2). Undetectability of Wat follows from the undetectability of Steg.

[9] Without loss of generality, we consider a binary token alphabet. This assumption is standard in the watermarking literature; see [CG24, Section 7.1] for a detailed discussion.

[10] Because robust block steganography schemes must be expanding in the message, the signature length must be smaller than the resulting block size, i.e., $|\sigma| \ll n$.

To verify a candidate text $\zeta \in \{0,1\}^{2n}$, the verification algorithm $\mathsf{Ver}(\mathsf{vk}, \zeta)$ first parses it into two blocks $\zeta_1, \zeta_2 \in \{0,1\}^n$. It then tries to decode a signature from the second block and check if it verifies with the first,

$$\mathsf{Ver}(\mathsf{vk}, \zeta) = \mathsf{DSS.Ver}(\mathsf{DSS.pk}, \zeta_1, \sigma'), \text{ where } \sigma' \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.dk}, \zeta_2) \ .$$

Notice that this approach is unforgeable in that it only verifies $\zeta = (\zeta_1, \zeta_2) \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$ when the first block ζ_1 *exactly* matches one of the blocks y_1 for which $\mathcal{A}$ saw a signature for; otherwise, $\mathcal{A}$ could break the unforgeability of DSS with the message ζ_1. However, the scheme provides no robustness. Concretely, suppose an adversary $\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$ queries its watermarking oracle and receives a response (y_1, y_2). If $\mathcal{A}$ modifies even a single bit of y_1 to produce ζ_1, the string (ζ_1, y_2) can no longer verify, despite being nearly identical to an observed response.

Robust signatures for both robustness and unforgeability. To overcome the limitation outlined above, we instead embed a *robust* signature into the generated text. Now, even if $\mathcal{A}$ modifies y_1 to produce ζ_1, so long as $\Phi(\zeta_1, y_1) = 1$, the signature—and therefore the watermark—can still verify.

Formally, let $\mathsf{RDS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ be a *robust* signature scheme for the same predicate Φ that Steg is robust to, and generate keys $(\mathsf{vk}, \mathsf{sk})$ as in the standard digital signature approach, but using $\mathsf{RDS.Gen}(1^\lambda)$ instead of $\mathsf{DSS.Gen}(1^\lambda)$. We generalize the approach above to generate rn bits for any $r \geq 2$. For the first block, $t = 1$, the algorithm $\mathsf{Wat}(\mathsf{sk}, x)$ invokes $y_1 \leftarrow \overline{Q}_n(x)$. For subsequent blocks $t \geq 2$, it updates the context to be the prompt and the text generated up to this point, $\pi = (x, y_1, \ldots, y_{t-1})$, and embeds the robust signature of the previous block:

$$\sigma_{t-1} \leftarrow \mathsf{RDS.Sign}(\mathsf{RDS.sk}, y_{t-1}), \quad y_t \leftarrow \mathsf{Steg.Embed}(\mathsf{Steg.sk}, \pi, \sigma_{t-1}) \ ,$$

The final output is the concatenation of blocks $(y_1, \ldots, y_r)$.

Watermark verification. We define watermark verification to operate on rn-bit strings for some $r \geq 2$. Given a candidate string $\zeta^* = (\zeta_1^*, \ldots, \zeta_r^*)$ composed of r blocks of n-bits, the verification algorithm $\mathsf{Ver}_r(\mathsf{vk}, \zeta^*)$ checks whether the *entire* chain of message-signature pairs verifies. Concretely, this means verifying that

$$\forall i \in [r-1], \ \mathsf{RDS.Ver}(\mathsf{RDS.pk}, \zeta_i^*, \sigma_i') = 1, \text{ where } \sigma_i' \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.dk}, \zeta_{i+1}^*) \ .$$

If RDS is additionally recoverable, we can define a corresponding recovery procedure $\mathsf{Rec}_r(\mathsf{vk}, \zeta^*)$ as follows.

$$\forall i \in [r-1], \ \xi_i \leftarrow \ \mathsf{RDS.Rec}(\mathsf{RDS.pk}, \zeta_i^*, \sigma_i'), \text{ where } \sigma_i' \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.dk}, \zeta_{i+1}^*) \ .$$

The procedure outputs a single string $(\xi_1, \ldots, \xi_{r-1})$ provided that none of the recovery attempts return $\perp$, else it outputs $\perp$.

Guarantees of our construction when $r = 2$. Consider the simple case $r = 2$. To see that verification is robust, let $\zeta^* = (\zeta_1^*, \zeta_2^*)$ be a candidate text produced by an efficient adversary $\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$. Say that ζ^* is approximately contained in some response $y \leftarrow \mathsf{Wat}_{\mathsf{sk}}(x)$ observed by $\mathcal{A}$, meaning that y contains two consecutive n-bit blocks (y_t, y_{t+1}) such that both $\Phi(\zeta_1^*, y_t) = 1$ and $\Phi(\zeta_2^*, y_{t+1}) = 1$. By the Φ-robustness of Steg, we can decode the signature σ_t of y_t from ζ_2^*. Moreover, by the Φ-robustness of RDS, the pair (ζ_1^*, σ_t) will pass verification. Similarly, in this setting, the recovery algorithm outputs the first response block y_t.

Moreover, to see the watermark is unforgeable, notice that unforgeability of the digital signature says that if $\mathsf{Ver}_2(\mathsf{vk}, (\zeta_1^*, \zeta_2^*)) = 1$ (or $\mathsf{Rec}_2(\mathsf{vk}, (\zeta_1^*, \zeta_2^*)) = y_t$) then ζ_1^* must be Φ-close to one of the signed blocks y_t observed by $\mathcal{A}$. We formalize our robustness and unforgeability guarantees in Theorem 6.1, with recoverability stated in Corollary 6.4.

Our construction does, however, exhibit a subtle gap. We can guarantee that verification succeeds when *both blocks* (ζ_1^*, ζ_2^*) are close to back-to-back response blocks (y_t, y_{t+1}), whereas we can only ensure that successful verification implies the *first block* ζ_1^* is close to a single response block y_t. This asymmetry arises because we embed signatures directly into subsequent blocks: to decode the signature and ensure verification succeeds, both blocks must be close. Successful RDS verification, on the other hand, only certifies the closeness of the underlying message (the first block). We are optimistic that future strategies for embedding messages into AI-generated content could help narrow this gap.

Guarantees when $r > 2$. In Theorem 6.2, we show that Ver_r satisfies a stronger unforgeability guarantee when $r > 2$. Specifically, we show that if $\mathsf{Ver}_r(\mathsf{vk}, (\zeta_1^*, \ldots, \zeta_r^*)) = 1$, then not only is each block ζ_i^* for $i \in [r-1]$ Φ-close to *some* response block, but moreover the blocks $(\zeta_1^*, \zeta_2^*, \ldots, \zeta_{r-1}^*)$ are each Φ-close to the aligned blocks of a *consecutive chain* $(y_t, y_{t+1}, \ldots, y_{t+r-2})$ present in a *single response* y.

This notion of closeness is captured by the every-block-close lifting of Φ (Definition 3.2). In particular, we obtain an unforgeability guarantee of the following form: $(\zeta_1^*, \ldots, \zeta_r^*)$ verifies only if for there is some watermarked response y such that $\mathsf{EBC}[\Phi, n]((\zeta_1^*, \ldots, \zeta_{r-1}^*), y) = 1$. We obtain a matching robustness guarantee: verification succeeds whenever there exists a response y such that $\mathsf{EBC}[\Phi, n]((\zeta_1^*, \ldots, \zeta_r^*), y) = 1$.

Generalizing verification to arbitrary-sized strings. In the discussion above, the algorithms Ver_r and Rec_r operate on inputs of fixed length $\zeta^* \in \{0,1\}^{rn}$. Semantically, however, it is more natural for watermark verification to operate on inputs of arbitrary length. In particular, given $\zeta \in \{0,1\}^*$, verification should succeed if and only if some sufficiently long *substring* ζ^* of ζ is approximately contained in a watermarked response. For example, if a student copies only a portion of an essay from a watermarked response, the verification algorithm should still verify the watermark when given the full essay.

To capture this more semantically meaningful notion, our formal security definitions (Definitions 3.4 and 3.5) differ slightly from the informal ones given in this overview. As a concrete example, for a closeness predicate Φ and a substring-length $\ell : \mathbb{N} \to \mathbb{N}$, we define public-key (Φ, ℓ)-unforgeability as follows:

$$\Big(\mathsf{Ver}(\mathsf{vk}, \zeta) = 1\Big) \implies \left(\begin{array}{l} \exists \zeta^* \preceq \zeta \text{ of size } \ell(\lambda), \\ \exists (x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}) \text{ such that } \Phi(\zeta^*, y) = 1 \end{array} \right),$$

where $\zeta^* \preceq \zeta$ denotes that ζ^* is a substring of ζ.

Defining robustness and unforgeability in this way also has the technical advantage of addressing the gap noted in Remark 1.6, as it allows each security property to be stated over substrings of different lengths. In particular, the construction outlined above can be easily adapted to achieve $(\mathsf{EBC}[\Phi, n], rn)$-robustness and $(\mathsf{EBC}[\Phi, n], (r-1)n)$-unforgeability. This can be done by defining a verifier $\mathsf{Ver}(\mathsf{vk}, \zeta)$ that takes as input an arbitrary-length string $\zeta \in \{0,1\}^*$ and performs r-block verification $\mathsf{Ver}_r(\mathsf{vk}, \zeta^*)$ on *every* substring $\zeta^* \preceq \zeta$ of length rn, accepting if *any* such substring verifies. We formalize this algorithm in Figure 6.

2.3 Discussion

We view our work as an initial step toward a broader line of investigation. Our framework treats robustness and undetectability as the traditional goals of watermarking, while focusing on introducing and achieving the new properties of unforgeability and recoverability without sacrificing the former. We demonstrate its effectiveness by instantiating language watermarking schemes for the every-block-close lifting of the Hamming predicate.

The resulting notion of robust signatures is versatile. Our general construction paradigm suggests avenues for improvement and indicates that our ideas can be extended to other modalities of generative AI. We address some concerns, as well as potential applications of our framework to other settings.

Why Ideal PRCs do not offer unforgeability? Ideal pseudorandom codes (PRCs) [AACD+24] do provide sharp decoding guarantees: any (adversarially chosen) ζ that is sufficiently far from all codewords decodes to $\perp$. At first glance, this suggests that ideal PRCs alone might yield secret key unforgeable watermarks. Indeed, a PRC can be seen as a *secret key* robust and unforgeable watermarking scheme for the *uniform* language model, where each bit is equally likely, as text generated by the model is indistinguishable from PRC codewords.

For general language models, however, embedding PRC codewords into text introduces errors that break these guarantees. When the embedding error is large, a string ζ that is close to some watermarked response y may be far from the PRC codeword embedded in y, and a string ζ that is far from all valid responses y may still be close to a codeword. Therefore, even if ζ is far from all responses y, if the embedding errors are large it can still verify as watermarked. Typically, this error is bounded by making an assumption on the entropy of the language models: a lower bound on entropy yields an upper bound on the errors incurred when embedding PRC codewords into text.

In contrast, our unforgeability guarantees are independent of the errors introduced by embedding, enabling protection against false positives without imposing entropy assumptions. Moreover, our framework is compatible with *any* block steganography scheme, regardless of its guarantees against false positives, which allows us to achieve a watermarking scheme that is both public key robust and public key unforgeable. Finally, our approach supports recoverability without requiring access to the underlying language model or the original generation prompt.

About our Hamming guarantees. Our final guarantee is a watermarking scheme for the every-block-close lifting of the Hamming predicate. We acknowledge that Hamming distance between binary token strings may not capture a wide class of watermark removal attacks. However, our general construction paradigm means any new improvement to block steganography and PPHs, e.g. improved error rates or support for different types of errors, will translate to improved unforgeable and recoverable watermarking. For example, we believe that robust signatures for a predicate we call *GapEdit*, which outputs 1 when the edit distance between two strings is small, 0 when it is very large, and is undefined for intermediate distances, are within reach. We expect such signatures to be attainable by instantiating our construction with a secure sketch for edit-distance [DORS08].

Post-hoc vs. ad-hoc steganography. We leverage ad-hoc steganography, where the message is embedded into the text during generation, to encode robust signatures directly into model outputs. In contrast, post-hoc steganography schemes, such as Google's SynthID image watermark [GBSS+25], embed the message after generation. That is, the embedder first generates $y \leftarrow \overline{Q}(x)$ normally and then post-processes it to obtain $y' \approx y$, where a given message is embedded into y'.

Post-hoc steganography schemes cannot guarantee undetectability, but often maintain, at least empirically, that the modified content y' is "perceptually" similar to the original response y.

Robust signatures are a natural choice for the message to be embedded using a post-hoc steganography scheme. One can generate $y \leftarrow \overline{Q}(x)$, compute its robust signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, y)$, and embed σ into $y' \approx y$ using a post-hoc watermarking scheme. Later, the signature can be decoded from $\zeta \approx y'$ and verified via robust signature verification $\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma)$. Note, however, that some robustness may be lost due to the modifications introduced during post-hoc steganography, since σ is the robust signature of the original response y, and the message ζ is close to the post-processed output $y' \approx y$. In this paper, we instead focus on ad-hoc steganography due to its stronger theoretical guarantees.

Robust and unforgeable image watermarking. Fairoze et al. [FOVJ+25] argue that constructing post-hoc watermarking schemes for *image models* that are both robust and unforgeable may be difficult. While our work focuses on language models, our results in fact suggest the opposite conclusion under the right abstraction.

In their work, they assume the existence of an embedding function $g : \mathbb{R}^n \to \mathbb{R}^m$ with $m < n$ that compresses images and preserves visual similarity: namely, that two images $y, y' \in \mathbb{R}^n$ are visually similar if and only if $g(y)$ and $g(y')$ are close under a chosen distance metric (they consider cosine distance). They show that by combining such an embedding with standard digital signatures and a post-hoc steganography scheme for images, they can construct a robust and unforgeable post-hoc watermarking scheme for image models. At a high level, their unforgeability guarantee is as follows: if a watermark is detected in an adversarially-chosen image ζ, then the embedding $g(\zeta)$ must be close to the embedding of some watermarked image $g(y)$ observed by the adversary.

However, they subsequently show that the assumed embedding function g may be overly idealized. In particular, they evaluate several candidate embedding functions (e.g., the forward pass of a trained deep network) and observe that these embeddings fail to satisfy the desired property. Specifically, it is easy to efficiently find images y and y' that are visually dissimilar, yet whose embeddings $g(y)$ and $g(y')$ are close in cosine distance. As a consequence, the unforgeability guarantee becomes vacuous: even if the underlying watermarking scheme is unforgeable, an adversary can produce an image that is visually far from all watermarked images, yet still verifies as watermarked because its embedding is close to the embedding of a previously observed image.

In this work, we overcome this issue for *language models* by leveraging PPHs. Rather than compressing text via a heuristic, similarity-preserving embedding g, we hash it using a PPH, which enables evaluating the similarity of the original texts directly (even if the hashes themselves may not be close under any distance metric). The adaptive security of PPH rules out attacks of the form outlined above.

In principle, this approach also extends to post-hoc *image* watermarking by using a PPH instead of the embedding function g in the construction of [FOVJ+25, Section 5.3]. However, Hamming PPHs are not especially compelling in this setting: one could hash the binary representations of pixels of the images, but this would effectively deem two images visually similar if and only if the Hamming distance between the binary representation of their pixels is small. PPHs for ℓ_2-distance do exist under the recently proposed contracting hypergrid vector assumption [BRVV25], when the input space consists of bounded integer vectors. If this bound is set to 256, then one could hash raw (unnormalized) pixel values so that two images are deemed similar iff their pixels are close in ℓ_2-distance. Likewise, one could hash the latent vector used to generate an image during diffusion, which is often interpreted as capturing the image's semantic meaning. Unfortunately, such PPH constructions may not be practical, particularly when the coordinate bound is as large as 256,

highlighting the need for further improvements.

3 Watermarking definitions

Notation. For strings $x, y \in \{0,1\}^*$, we write $x \preceq y$ to denote that x is a *substring* of y. We write (x, y) for their concatenation and $|y|$ for the length of y. We write $y[i]$ for the ith bit of y, and $y[i:j]$ for $i < j$ to denote the substring $(y[i], y[i+1], \ldots, y[j-1])$ (excluding the jth bit).

For $q \in \mathbb{N}$, we denote $[q] = \{1, 2, \ldots, q\}$. For a set X, we write $x \leftarrow X$ to denote that x is a random variable sampled uniformly from X. For a (probabilistic) algorithm $\mathcal{A}$ and a (possibly random) input Y, we write $x \leftarrow \mathcal{A}(Y)$ to denote that x is a random variable distributed according to the output of $\mathcal{A}$ when run on input Y.

Let $\lambda \in \mathbb{N}$ be a security parameter. We write $n = \mathrm{poly}(\lambda)$ if $n : \mathbb{N} \to \mathbb{N}$ is a polynomial function. When λ is clear from context, we simply write n for $n(\lambda)$. For a function $f : \mathbb{N} \to \mathbb{R}$, we write $f(\lambda) \leq \mathrm{negl}(\lambda)$ to indicate that f is bounded above by some negligible function.

For $p \in [0,1]$, we write $\mathsf{Ber}(p)$ for the Bernoulli distribution on $\{0,1\}$ with expectation p.

Language models. A language model $Q : \{0,1\}^* \to [0,1]$ maps a token sequence to the probability that the next token is 1. Given a prompt $x \in \{0,1\}^*$, we obtain a *response* $u = (u[1], u[2], \ldots) \leftarrow \overline{Q}(x)$ by autoregressive sampling initialized with x. Formally, each bit $u[j]$ of the response is sampled as

$$u[j] \leftarrow \mathsf{Ber}(Q(x, u[1:j])) \ .$$

(x, u) is a single prompt-response pair. For $\ell \in \mathbb{N}$ we write $\overline{Q}_\ell(x)$ for the distribution obtained from autoregressively sampling the next ℓ bits.

Transcripts. We consider adversaries that interact with oracles M by submitting inputs $x \in \{0,1\}^*$ and receiving responses $y \in \{0,1\}^*$. For oracles that accept multiple inputs, x may be viewed as a concatenation of these inputs. For a (randomized) adversary $\mathcal{A}$, we denote by $\mathsf{Trans}(\mathcal{A}^{\mathsf{M}})$ the *transcript* of this interaction, i.e., the random variable describing the set of all input-output pairs observed by $\mathcal{A}$.

Blocks. For a string $y \in \{0,1\}^*$ and a block size $n \in \mathbb{N}$, we define the n-bit block parsing of y as

$$\mathsf{Blocks}(y) \ = \ (y_1, y_2, \ldots, y_{\lfloor |y|/n \rfloor}) \, ,$$

where each $y_t \in \{0,1\}^n$ is the substring $y[(t-1)n+1 : tn+1]$. Any trailing bits that do not form a full block are discarded. For convenience, we sometimes write "an n-sized block y_t of y", or just y_t when n is clear from context. In such cases, we mean: let y_t denote the t-th entry in the sequence given by $\mathsf{Blocks}(y)$.

Predicates. We use predicates to formalize "closeness" between strings. A *predicate* is a Boolean-valued function $\Phi : \mathcal{X} \to \{0,1\}$ defined on some domain $\mathcal{X}$. Throughout this paper, when we quantify over predicates, we implicitly restrict to predicates that are efficiently computable. A canonical example is the Hamming predicate, which checks whether two strings differ in at most a δ-fraction of positions.

Definition 3.1 (Hamming)**.** *For any $\delta \in [0,1]$, we define the Hamming predicate Ham_δ as follows. For any strings $y, y' \in \{0,1\}^*$,*

$$\mathsf{Ham}_\delta(y, y') = (|y| = |y'| \ \wedge \ \|y - y'\|_0 \leq \delta|y|) \ .$$

For a predicate Φ with input domain $\mathcal{X} = \{0,1\}^* \times \{0,1\}^*$ and strings $y, y' \in \{0,1\}^*$, we say that y is Φ-close to y' if $\Phi(y, y') = 1$, and Φ-far from y' if $\Phi(y, y') = 0$.

Every block close predicate. To state the guarantees of our watermarking construction, it will be useful to have in mind a notion of *every-block-closeness* for a predicate Φ. Intuitively, this notion *lifts* Φ, which originally may only compare two equal-sized strings of size n, to a predicate on sequences of blocks: it checks whether the first input is composed of consecutive n-bit blocks that are Φ-close to a sequence of consecutive n-bit blocks contained in the second input.

Definition 3.2 (Every-block close)**.** *For a predicate Φ and block size $n \in \mathbb{N}$, we define the predicate $\mathsf{EBC}[\Phi, n]$ (every-block-close) as*

$$\mathsf{EBC}[\Phi, n](\zeta^*, y) : \begin{pmatrix} \textit{Check that } \exists r \in \mathbb{N} \textit{ such that } |\zeta^*| = rn. \textit{ Output 0 if not.} \\ \textit{Parse } (\zeta_1^*, \ldots, \zeta_r^*) = \mathsf{Blocks}(\zeta^*), \\ (y_1, \ldots, y_{\lfloor |y|/n \rfloor}) = \mathsf{Blocks}(y) \textit{ into } n\textit{-bit blocks.} \\ \textit{Output 1 iff } \exists t \in \mathbb{N} \textit{ such that } \forall j \in [r], \ \Phi(\zeta_j^*, y_{t+j-1}) = 1. \end{pmatrix}$$

Recall Figure 2, which illustrates $\mathsf{EBC}[\Phi, n]$. The predicate $\mathsf{EBC}[\Phi, n]$ is generally asymmetric: the first input ζ^* is treated as a sequence of r consecutive n-bit blocks, while the second input y may be longer. The predicate searches for a sequence of r consecutive blocks *within* y that are blockwise Φ-close to ζ^*. When ζ^* and y have the same length, the predicate becomes symmetric (assuming Φ is symmetric) and reduces to a simple blockwise comparison, checking that their aligned blocks are Φ-close.

3.1 Watermarking

We start by introducing the syntax of a *watermarking scheme* for a language model Q, a tuple of algorithms that act as the interface for generating and verifying watermarked outputs. We require that watermarking schemes satisfy a strong notion of quality preservation: undetectability [GZS24]. This guarantees that watermarked outputs from Wat are indistinguishable from standard autoregressive samples from Q to any efficient adversary that can choose input prompts adaptively.

Definition 3.3 (Watermarking)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter. A watermarking scheme for a language model Q and token set $\{0,1\}$ is a tuple of polynomial time algorithms $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$ satisfying,*

- $\mathsf{Gen}(1^\lambda)$: *takes in a security parameter λ and outputs verification and secret keys $(\mathsf{vk}, \mathsf{sk})$.*
- $\mathsf{Wat}(\mathsf{sk}, x)$: *takes in the secret key sk and a prompt $x \in \{0,1\}^*$ and outputs a response $y \in \{0,1\}^*$.*
- $\mathsf{Ver}(\mathsf{vk}, \zeta)$: *takes in the verification key vk and input text $\zeta \in \{0,1\}^*$ and outputs a verification bit $b \in \{0,1\}$.*

A recoverable watermarking scheme additional requires the following PPT algorithm Rec,

- $\mathsf{Rec}(\mathsf{vk}, \zeta)$: *takes in the verification key* vk *and input text* $\zeta \in \{0,1\}^*$ *and outputs a list* $\mathcal{L} \subseteq \{0,1\}^*$ *of recoveries or a failure symbol* $\perp$.

Undetectability *For every PPT distinguisher* $\mathcal{A}$*,*

$$\left| \Pr_{(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)} \left[\mathcal{A}^{Q, \mathsf{Wat}_{\mathsf{sk}}}(1^\lambda) = 1 \right] - \Pr \left[\mathcal{A}^{Q, \overline{Q}}(1^\lambda) = 1 \right] \right| \leq \mathrm{negl}(\lambda) \ .$$

We now define our security notions for watermarking schemes. Throughout this section, we present the *secret key* variants, where the adversary has oracle access to a verification oracle $\mathsf{Ver}_{\mathsf{vk}}$. In Remark 3.8, we describe the corresponding public key variants, obtained by simply giving the adversary explicit access to vk.

Robustness requires that it is hard for an efficient adversary—even with access to both the watermarking oracle and the verification oracle—to produce a string ζ that contains a ℓ_{rb}-sized substring $\zeta^* \preceq \zeta$ which is Φ-*close* to one of the watermarked responses $y \leftarrow \mathsf{Wat}(\mathsf{sk}, x)$ it observed, yet for which ζ fails to verify.

Definition 3.4 (Robustness)**.** *For any closeness predicate* Φ *and any substring size* $\ell_{\mathsf{rb}} = \mathrm{poly}(\lambda)$*, a watermarking scheme* $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$ *is secret key* $(\Phi, \ell_{\mathsf{rb}})$-robust *if for any PPT algorithm* $\mathcal{A}$ *outputting* $\zeta \in \{0,1\}^*$*,*

$$\Pr \left[\begin{array}{rl} (\mathsf{vk}, \mathsf{sk}) & \leftarrow \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}, \mathsf{Ver}_{\mathsf{vk}}}(1^\lambda) \end{array} : \begin{array}{r} \Big(\exists \zeta^* \preceq \zeta, (x,y) \in \Pi \text{ such that} \\ \Phi(\zeta^*, y) = 1 \wedge \ |\zeta^*| = \ell_{\mathsf{rb}} \Big) \\ \wedge \ \mathsf{Ver}(\mathsf{vk}, \zeta) = 0 \end{array} \right] \leq \mathrm{negl}(\lambda) \ ,$$

where Π *denotes the transcript* $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$.

We include the substring requirement, $|\zeta^*| = \ell_{\mathsf{rb}}$, for two reasons. First, we cannot guarantee that the watermark is verifiable in arbitrarily small ζ, so this requirement imposes a lower bound on the size of ζ. Second, it allows us to capture adversaries who approximately copy a portion of their output, ζ^*, from the watermarked model, then insert arbitrary text around it. The substring size specifies exactly how much the text must be copied for the watermark to verify, regardless of the surrounding content.

Unforgeability complements robustness by requiring that it is hard for an efficient adversary to produce a string ζ that verifies as watermarked, but for which ζ contains *no* ℓ_{uf}-sized substring $\zeta^* \preceq \zeta$ that is Φ-close to any watermarked response y observed by the adversary. Another interpretation of unforgeability is that if all ℓ_{uf}-sized substrings $\zeta^* \preceq \zeta$ are *far* from all of the watermarked responses y seen by the adversary, then verification must fail.

Definition 3.5 (Unforgeability)**.** *For any closeness predicate* Φ *and any substring size* $\ell_{\mathsf{uf}} = \mathrm{poly}(\lambda)$*, a watermarking scheme* $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$ *is secret key* $(\Phi, \ell_{\mathsf{uf}})$-unforgeable *if for any PPT algorithm* $\mathcal{A}$ *outputting* $\zeta \in \{0,1\}^*$*,*

$$\Pr \left[\begin{array}{rl} (\mathsf{vk}, \mathsf{sk}) & \leftarrow \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}, \mathsf{Ver}_{\mathsf{vk}}}(1^\lambda) \end{array} : \begin{array}{r} \mathsf{Ver}(\mathsf{vk}, \zeta) = 1 \\ \wedge \neg \Big(\exists \zeta^* \preceq \zeta, (x,y) \in \Pi \text{ such that} \\ \Phi(\zeta^*, y) = 1 \ \wedge \ |\zeta^*| = \ell_{\mathsf{uf}} \Big) \end{array} \right] \leq \mathrm{negl}(\lambda) \ ,$$

where Π *denotes the transcript* $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$.

These definitions provide security for binary watermark verification. However, we may wish to go further and track the edits applied to content by recovering the original model outputs. *Recoverable watermarking schemes* provide exactly this capability. Robustness for a recoverable watermarking scheme requires that no efficient adversary can produce a string ζ containing a sufficiently long substring $\zeta^* \preceq \zeta$ that is Φ-close to some watermarked response y, yet for which the recovery algorithm does not recover a substring of that response $\xi \preceq y$. This recovered substring ξ must also be similar to the copied ζ^* in size (up to an allowed loss).

Definition 3.6 (Robust recoverability)**.** *For any closeness predicate Φ, substring size $\ell_{\mathsf{rec}} = \mathrm{poly}(\lambda)$, and loss size $\ell_{\mathsf{loss}} = \mathrm{poly}(\lambda)$, a* recoverable *watermarking scheme* $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Rec})$ *is secret key* $(\Phi, \ell_{\mathsf{rec}}, \ell_{\mathsf{loss}})$-robust *if for any PPT algorithm $\mathcal{A}$ outputting $\zeta \in \{0,1\}^*$,*

$$\Pr\left[\begin{array}{rcl} (\mathsf{vk},\mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow & \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}},\mathsf{Rec}_{\mathsf{vk}}}(1^\lambda) \\ \mathcal{L} & \leftarrow & \mathsf{Rec}(\mathsf{vk},\zeta) \end{array} : \begin{array}{r} \Big(\exists \zeta^* \preceq \zeta, (x,y) \in \Pi \text{ such that} \\ \Phi(\zeta^*, y) = 1 \ \wedge \ |\zeta^*| \geq \ell_{\mathsf{rec}}\Big) \wedge \\ \mathcal{L} \text{ does not contain a} \\ (|\zeta^*| - \ell_{\mathsf{loss}})\text{-sized substring of } y \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

where Π denotes the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$.

To ensure that recoveries are reliable, it is natural to additionally require that any recovered text truly originates from the model. This requirement is especially important in applications where recovered substrings are used for fine-grained attribution. One could rely on standard unforgeability, which guarantees that whenever the recovered list $\mathsf{Rec}(\mathsf{vk},\zeta)$ contains a string ξ, there exists *some* observed response y that is close to a substring $\zeta^* \preceq \zeta$. However, this does not ensure that the recovery ξ itself was genuinely observed by the adversary. Instead, we require *unforgeable recoverability*: no efficient adversary can produce text ζ such that $\mathsf{Rec}(\mathsf{vk},\zeta)$ recovers a string ξ which is either too small, or is *not* a substring of any response y that was approximately copied by the adversary.

Definition 3.7 (Unforgeable recoverability)**.** *For any closeness predicate Φ and any substring size $\ell_{\mathsf{rec}} = \mathrm{poly}(\lambda)$, a* recoverable *watermarking scheme* $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Rec})$ *is secret key* $(\Phi, \ell_{\mathsf{rec}})$-unforgeable *if for any PPT algorithm $\mathcal{A}$ outputting $\zeta \in \{0,1\}^*$,*

$$\Pr\left[\begin{array}{rcl} (\mathsf{vk},\mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow & \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}},\mathsf{Rec}_{\mathsf{vk}}}(1^\lambda) \end{array} : \begin{array}{r} \exists \xi \in \mathsf{Rec}(\mathsf{vk},\zeta) \text{ such that } \Big(|\xi| < \ell_{\mathsf{rec}}\Big) \\ \vee \neg\Big(\exists \zeta^* \preceq \zeta, (x,y) \in \Pi \text{ with} \\ \Phi(\zeta^*, y) = 1 \wedge \ |\zeta^*| = |\xi| \ \wedge \ \xi \preceq y\Big) \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

where Π denotes the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$.

Remark 3.8. *We presented the secret key variants of all our definitions, where the attacker receives oracle access to both $\mathsf{Wat}_{\mathsf{sk}}$ and $\mathsf{Ver}_{\mathsf{vk}}$. We also consider the corresponding public key variants, which are identical except that the attacker additionally receives the full verification key vk (rather than only black-box access to the verification oracle).*

Remark 3.9. *We do not include a loss term ℓ_{loss} in the definition of unforgeable recoverability, because our construction guarantees the existence of a substring $\zeta^* \preceq \zeta$ that has* exactly *the same length as the recovered string.*

Remark 3.10. *We define standard robustness to consider fixed-size substrings $|\zeta^*| = \ell_{\mathsf{rb}}$, but allow variable-size substrings $|\zeta^*| \geq \ell_{\mathsf{rec}}$ for recoverable robustness. One could additionally define a dynamic version of standard robustness, in which verification succeeds whenever there exists a close substring ζ^* with $|\zeta^*| \geq \ell_{\mathsf{rb}}$. Our construction in fact satisfies this property. However, we include it explicitly only in the definition of recoverability for two reasons:*

1. *It is meaningful there, as it allows us to capture that the size of the recovered output $\xi \in \mathcal{L}$ scales with the size of the approximate copy $\zeta^* \preceq \zeta$.*
2. *Standard robustness typically implies this behavior implicitly. In particular, consider $(\Phi, \ell_{\mathrm{rb}})$-robustness for a predicate Φ satisfying the following: for any $\zeta \in \{0,1\}^*$ with $|\zeta| \geq \ell_{\mathsf{rb}}$,*

$$\Phi(\zeta, y) = 1 \implies \exists \zeta^* \preceq \zeta \text{ with } |\zeta^*| = \ell_{\mathsf{rb}} \text{ and } \Phi(\zeta^*, y) = 1 \,,$$

Under this condition, standard robustness implies the dynamic notion of robustness. If ζ contains a substring ζ' with $|\zeta'| > \ell_{\mathsf{rb}}$ that is Φ-close to a response y, then by the above property there exists a further substring $\zeta^* \preceq \zeta' \preceq \zeta$ of size exactly ℓ_{rb} that is also Φ-close to y. $(\Phi, \ell_{\mathsf{rb}})$-robustness then guarantees that verification succeeds on ζ.

4 Preliminaries

Digital signatures. We recall the definition of (non-robust) digital signature schemes. Digital signature schemes allow a signer to associate a message with their publicly-verifiable signature. Parties without the signing key cannot forge valid signatures on messages that the signer has not signed. Here, we define digital signatures with *strong unforgeability* [BS07]. While standard unforgeability only prevents a forger from creating a valid signature for a new message, strong unforgeability goes further: it prevents the forger from producing a *new* valid signature, even for a message whose signature it has already seen.

Definition 4.1 (Digital signature scheme)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $n = \mathrm{poly}(\lambda)$ be the message size, and $k = \mathrm{poly}(\lambda)$ be the signature size. A digital signature scheme is a triple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ of PPT algorithms satisfying,*

- $\mathsf{Gen}(1^\lambda)$: *outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.*
- $\mathsf{Sign}(\mathsf{sk}, y)$: *takes in the secret key sk and a message $y \in \{0,1\}^n$ and outputs a signature $\sigma \in \{0,1\}^k$.*
- $\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma)$: *takes in the public key pk, a string $\zeta \in \{0,1\}^n$, and a signature $\sigma \in \{0,1\}^k$ and outputs a binary decision $b \in \{0,1\}$.*

Correctness: *For any $y \in \{0,1\}^n$,*

$$\Pr\left[\begin{array}{rl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma & \leftarrow \mathsf{Sign}(\mathsf{sk}, y) \end{array} \;:\; \mathsf{Ver}(\mathsf{pk}, y, \sigma) = 1 \right] = 1 \,,$$

Strong unforgeability: *For any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow \mathsf{Gen}(1^\lambda) \\ (\zeta, \sigma) & \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk}) \end{array} \;:\; \Big(\mathsf{Ver}(\mathsf{pk}, \zeta, \sigma) = 1\Big) \wedge \Big((\zeta, \sigma) \notin \Pi\Big) \right] \leq \mathrm{negl}(\lambda) \,,$$

where Π is the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}})$.

Collision resistant hash. The sharp sketch construction of Dodis et al. [DORS08] and property-preserving hash construction of Holmgren et al. [HLTW22] rely on the existence of collision resistant hash functions (CRHFs). We recall their definition here.

Definition 4.2 (Collision resistant hash)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter and let $n = \mathrm{poly}(\lambda)$ and $k = \mathrm{poly}(\lambda)$ be such that $k \leq n$. A collision resistant hash family $(\mathsf{Sample}, \{\mathcal{H}_\lambda\})$ consists of an efficient algorithm $h \leftarrow \mathsf{Sample}(1^\lambda)$ that samples a hash function $h \in \mathcal{H}_\lambda$ with $h : \{0,1\}^n \to \{0,1\}^k$. We require,*

Efficiency: *Given $x \in \{0,1\}^n$ and $h \leftarrow \mathsf{Sample}(1^\lambda)$, the function $h(x)$ can be computed in time $\mathrm{poly}(\lambda)$.*

Collision-resistance: *For any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} h & \leftarrow \mathsf{Sample}(1^\lambda) \\ (x, x') & \leftarrow \mathcal{A}(1^\lambda, h) \end{array} \;:\; \begin{array}{r} h(x) = h(x') \\ \wedge\, (x \neq x') \end{array}\right] \leq \mathrm{negl}(\lambda)\ .$$

Pseudorandom codes. Pseudorandom codes (PRCs) are error-correcting codes whose codewords are indistinguishable from random binary vectors [CG24]. We use ideal security (defined by Alrabiah et al. [AACD+24]), which means that even an adversary with *oracle* access to the encoder and decoder cannot distinguish between a PRC scheme and an "ideal" random code scheme, whose codewords are uniformly random and decoder simply responds with exact proximity queries to the underlying transcript. Ideal security thus captures both the pseudorandomness and robustness properties that PRCs are designed to achieve.

PRC encoding and decoding keys. In the PRC literature, the PRC encoding and decoding keys are often referred to as "public" and "secret" keys, respectively. However, in the watermarking setting, these roles are reversed: encoding is performed by the model provider, who holds the *secret* watermark key, while decoding is performed by users, who hold the *public* key and wish to verify the watermark. To avoid confusion, we refer to the PRC keys simply as the encoding and decoding keys, denoted ek and dk, respectively.

Definition 4.3 (Pseudorandom codes)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter, $k = poly(\lambda)$ be the message size, and $n = \mathrm{poly}(\lambda)$ be the block size. A PRC is a triple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of PPT algorithms satisfying,*

- $\mathsf{Gen}(1^\lambda)$: *outputs encoding and decoding keys $(\mathsf{ek}, \mathsf{dk})$.*
- $\mathsf{Enc}(\mathsf{ek}, \sigma)$: *takes in the encoding key ek and a message $\sigma \in \{0,1\}^k$ and outputs a codeword $y \in \{0,1\}^n$.*
- $\mathsf{Dec}(\mathsf{dk}, \xi)$: *takes in the decoding key dk and a string $\xi \in \{0,1\}^n$ and outputs message or decoding failure $\sigma \in \{0,1\}^k \cup \{\perp\}$.*

The information rate of the PRC is $\frac{k}{n}$.

Ideal security captures the core properties that PRCs are designed to satisfy. It asserts that no PPT adversary can distinguish between a real-world interaction with the PRC and an ideal-world interaction. In the real world, the adversary interacts with the PRC encoder and decoder via oracle access. In the ideal world, the adversary receives a fresh uniformly random string for each encoding query, and the decoder maintains a complete transcript of all past encodings, decoding only those

strings that are sufficiently close to previously generated ones. This notion thus captures both pseudorandomness and robustness within a single definition. It is formalized through two security games, $\mathcal{G}^{\mathsf{real}}$ and $\mathcal{G}^{\mathsf{ideal}}$.

$\underline{\mathcal{G}^{\mathsf{real}}_{\lambda}(\mathcal{A})}$:

1. The challenger samples $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PRC.Gen}(1^{\lambda})$.
2. The adversary makes encoding and decoding queries.
   - For each encoding query σ, the challenger responds with $y \leftarrow \mathsf{PRC.Enc}(\mathsf{ek}, \sigma)$.
   - For each decoding query ξ, the challenger responds with $\sigma' \leftarrow \mathsf{PRC.Dec}(\mathsf{dk}, \xi)$.
3. The adversary returns a decision bit $b \in \{0, 1\}$.

$\underline{\mathcal{G}^{\mathsf{ideal}}_{\delta,\lambda}(\mathcal{A})}$:

1. The challenger initializes a transcript $\Pi = \emptyset$.
2. The adversary makes encoding and decoding queries.
   - For each encoding query σ, the challenger responds with a fresh random codeword $y \leftarrow \{0, 1\}^n$ and adds the entry (σ, y) to the transcript Π.
   - For each decoding query ξ, the challenger responds with a random message σ satisfying $(\sigma, y) \in \Pi$ for some y such that $\mathsf{Ham}_{\delta}(y, \xi) = 1$. If no such σ exists, then the challenger responds with $\perp$.
3. The adversary returns a decision bit $b \in \{0, 1\}$.

Definition 4.4 (Ideal security)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter and $\delta \in (0, \frac{1}{2})$ be the error-rate. A pseudorandom code* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ideal *with error-rate δ if, for any PPT adversary $\mathcal{A}$,*

$$\left|\Pr\left[\mathcal{G}^{\mathsf{real}}_{\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathcal{G}^{\mathsf{ideal}}_{\delta,\lambda}(\mathcal{A}) = 1\right]\right| \leq \mathrm{negl}(\lambda) \ .$$

Alrabiah et al. [AACD+24] construct ideal pseudorandom codes with constant error rate $\delta \in (0, \frac{1}{4})$ while achieving a constant information rate $\rho \in (0, 1)$, assuming the subexponential hardness of the exact learning parity with noise problem (LPN) [JKPT12].

Theorem 4.5 ([AACD+24, Corollary 5])**.** *Assume the subexponential hardness of exact LPN [AACD+24, Assumption 1]. For any constant $\delta \in (0, \frac{1}{4})$, there exists a constant $\rho \in (0, 1)$ such that there exists an ideal PRC with error-rate δ and information rate $\rho + o(1)$.*

4.1 Property-preserving hash

Property preserving hash functions (PPHs) [BLV19] allow efficient computation of a predicate Φ on two strings using only their compressed hashes, even when those inputs are chosen adversarially with full knowledge of the hash.

Definition 4.6 (Property-preserving hash)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, let Φ be any predicate, and let $n = \mathrm{poly}(\lambda)$ and $k = \mathrm{poly}(\lambda)$ with $k \leq n$ be the input and hash size respectively. A PPH scheme for Φ is a triple of PPT algorithms* $(\mathsf{Sample}, \mathsf{Hash}, \mathsf{Eval})$ *satisfying*

- $\mathsf{Sample}(1^\lambda)$: *samples a hash function h from the family of functions $\mathcal{H}_\lambda$.*
- $\mathsf{Hash}(h, x)$: *takes in the hash description h and a string $x \in \{0,1\}^n$ and outputs a hash $z \in \{0,1\}^k$.*
- $\mathsf{Eval}(h, z, z')$: *takes in the hash description h and strings $z, z' \in \{0,1\}^k$ and outputs an evaluation $b \in \{0,1\}$.*

Direct-access robustness: *For any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} h & \leftarrow \mathsf{Sample}(1^\lambda) \\ (x, x') & \leftarrow \mathcal{A}(1^\lambda, h) \end{array} : \mathsf{Eval}(h, h(x), h(x')) \neq \Phi(x, x')\right] \leq \mathrm{negl}(\lambda) \ .$$

We use the notation $h(x) := \mathsf{Hash}(h, x)$.

Prior works [FS21; FLS22; HLTW22] have constructed PPHs for the Hamming predicate Ham_δ. In particular, assuming the existence of a special type of homomorphic CRHF, the construction of Holmgren et al. [HLTW22] achieves constant-factor compression of the input when the error rate is constant.

Theorem 4.7 ([HLTW22, Corollary 6.7])**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and let $n = \mathrm{poly}(\lambda)$ denote the input size. Suppose there exists a homomorphic CRHF with output size $\ell = \ell(\lambda)$ that is collision-resistant over $\{-1, 0, 1\}^n$ [HLTW22, Definition 6.1]. Then, for any constants $\delta \in (0, 1/2)$ and $\rho > H_3^{\mathsf{BZ}}(\delta) \cdot \log_2 3$, where $H_3^{\mathsf{BZ}}(\cdot)$ refers to the Blokh-Zyablov bound [HLTW22, Fact 2.12], there exists a PPH for Ham_δ with hash size $\rho \cdot n + \ell$.*

Remark 4.8 (Blokh-Zyablov)**.** *The Blokh-Zyablov bound [BZ82] is what best known constructions of efficiently list-decodable codes achieve [GR08]. It is a longstanding open problem in coding theory to match the Hamming bound $H(\delta)$ with efficient codes. Note that $\log \binom{n}{\delta n} = nH(\delta) + o(n)$.*

Remark 4.9. *Theorem 4.7 relies on a specialized form of homomorphic CRHF, which can be constructed under either the shortest integer solution (SIS) assumption or the discrete logarithm assumption. Later, we instantiate a slightly weaker hash family that relies on standard CRHFs, but still suffices for our purposes (Theorem 5.6).*

4.2 Block steganography

We use *steganography* as a tool in our watermarking constructions. Steganography is similar to watermarking but differs in three key respects. First, a steganography scheme enables messages to be robustly embedded in and decoded from generated text, rather than allowing only binary verification. Second, we only require correctness and robustness from steganography schemes, with no guarantees on soundness, unforgeability, or recoverability (i.e., no bound on false positives). Third, we consider *block steganography*, which produces a single, size n block per invocation, whereas a full watermarking scheme generates an entire response, which could be arbitrarily long. This distinction is not significant: as shown below, a block-steganography scheme can be used to autoregressively generate a full response. The block interface is simply more convenient for later constructions.

Definition 4.10 (Block steganography)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $k = poly(\lambda)$ be the message size, and $n = \mathrm{poly}(\lambda)$ be the block size. A block steganography scheme for a language model Q and token set $\{0,1\}$ is a tuple of polynomial time algorithms $(\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$ satisfying,*

- $\mathsf{Gen}(1^\lambda)$: *takes in a security parameter λ and outputs verification and secret keys $(\mathsf{vk}, \mathsf{sk})$.*

- $\mathsf{Embed}(\mathsf{sk}, \pi, m)$: *takes in the secret key* sk*, a context* $\pi \in \{0,1\}^*$*, and a message* $m \in \{0,1\}^k$ *and outputs a block* $y \in \{0,1\}^n$*.*
- $\mathsf{Dec}(\mathsf{vk}, \zeta)$: *takes in the verification key* vk *and input text* $\zeta \in \{0,1\}^n$ *and outputs a decoded message or a failure symbol* $m \in \{0,1\}^k \cup \{\perp\}$*.*

Undetectability *For every PPT distinguisher* $\mathcal{A}$*,*

$$\left| \Pr_{(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)} \left[\mathcal{A}^{Q, \mathsf{Embed}_{\mathsf{sk}}}(1^\lambda) = 1 \right] - \Pr \left[\mathcal{A}^{Q, \overline{Q}_n}(1^\lambda) = 1 \right] \right| \leq \mathrm{negl}(\lambda) \ ,$$

where both $\mathsf{Embed}_{\mathsf{sk}}$ *and* $\overline{Q}_n$ *take in contexts* $\pi \in \{0,1\}^*$ *and message* $m \in \{0,1\}^k$ *as input, but* $\overline{Q}_n$ *ignores the message:* $\overline{Q}_n(\pi, m) := \overline{Q}_n(\pi)$*.*

Robustness for block steganography is similar to robustness for watermarking, but allows decoding a message m from an input ζ that is close to an embedding y of that message.

Definition 4.11 (Block Robustness)**.** *For any closeness predicate* Φ*, a block steganography scheme* $(\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$ *with block size* $n = \mathrm{poly}(\lambda)$ *is* secret-key Φ-robust *if for any PPT algorithm* $\mathcal{A}$ *outputting* $\zeta \in \{0,1\}^n$*,*

$$\Pr \left[\begin{array}{rcl} (\mathsf{vk}, \mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow & \mathcal{A}^{\mathsf{Embed}_{\mathsf{sk}}, \mathsf{Dec}_{\mathsf{vk}}}(1^\lambda) \end{array} \ : \ \begin{array}{c} \Big(\exists (\pi, m, y) \in \Pi \text{ such that} \\ \Phi(\zeta, y) = 1 \wedge \ \mathsf{Dec}(\mathsf{vk}, \zeta) \neq m \Big) \end{array} \right] \leq \mathrm{negl}(\lambda) \ ,$$

where Π *denotes the transcript* $\mathsf{Trans}(\mathcal{A}^{\mathsf{Embed}_{\mathsf{sk}}})$*.*

Remark 4.12. *Public-key* Φ*-robustness for block steganography is defined identically to the secret key variant, except the adversary receives the verification key* vk*.*

Robust watermarking from block steganography. Many existing watermarking schemes are explicitly or implicitly based on block steganography [CG24; GM24; FGJM+25]. Below, we describe a simple watermarking scheme $(\mathsf{Gen}, \overline{\mathsf{Wat}}, \overline{\mathsf{Ver}})$ that uses a steganography scheme $(\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$ in an autoregressive manner, embedding arbitrary messages (e.g., the message 1) until a special done token signals that the model should output the string. While this scheme guarantees robustness, it does not provide unforgeability or recoverability, which are the focus of our construction in Section 6.1.

Algorithm 1 $\overline{\mathsf{Wat}}$

input: $(1^\lambda, \mathsf{sk}, x)$
$t \leftarrow 1$
$y_1 \leftarrow \mathsf{Embed}(\mathsf{sk}, x, 1)$
while $\mathsf{done} \not\preceq y_t$ **do**
  $t \leftarrow t + 1$
  $y_t \leftarrow \mathsf{Embed}(\mathsf{sk}, (x, y_1, \ldots, y_{t-1}), 1)$
end while
return $y = (y_1, \ldots, y_{t-1}, y_t[< \mathsf{done}])$

Here, we use the shorthand $y_t[< \mathsf{done}]$ to denote the substring of y_t preceding the done token. The verification algorithm then examines every substring of size n in its input ζ, and accepts if any of them decode to a message other than $\perp$.

Algorithm 2 $\overline{\mathsf{Ver}}$

input: $(1^\lambda, \mathsf{vk}, \zeta)$
$n \leftarrow n(\lambda)$
for $i \in [|\zeta| - n + 1]$ **do**
  $m \leftarrow \mathsf{Dec}(\mathsf{vk}, \zeta[i : i + n])$
  if $m \neq \perp$ **then**
    return 1
  end if
end for
return 0

In this construction, each block y_t of a response $y \leftarrow \overline{\mathsf{Wat}}(\mathsf{sk}, x)$ is generated as an embedding $y_t \leftarrow \mathsf{Embed}(\mathsf{sk}, \pi, 1)$. Consequently, in $\overline{\mathsf{Ver}}(\mathsf{vk}, \zeta)$, as long as the input ζ contains an n-bit substring $\zeta^* \preceq \zeta$ that is Φ-close to some block y_t of a response y, the decoder $\mathsf{Dec}(\mathsf{vk}, \zeta^*)$ outputs 1 except with negligible probability. This, in turn, ensures that $\overline{\mathsf{Ver}}(\mathsf{vk}, \zeta)$ also outputs 1. Recall that this condition is captured exactly by the predicate $\mathsf{EBC}[\Phi, n](\zeta^*, y)$ (Definition 3.2), which, given ζ^* of size n, outputs 1 iff it is Φ-close to a block of y. We formalize this guarantee in Fact 4.13.

Fact 4.13. *Let $\lambda \in \mathbb{N}$ be the security parameter, $n = \mathrm{poly}(\lambda)$ be any block size, Φ be any closeness predicate, and Q be any language model. If $(\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$ is a block size n steganography scheme for Q with (secret or public key) Φ-robustness, then the watermarking scheme $(\mathsf{Gen}, \overline{\mathsf{Wat}}, \overline{\mathsf{Ver}})$ for Q satisfies (secret or public key) $(\mathsf{EBC}[\Phi, n], n)$-robustness.*

Proof. Undetectability follows immediately from the undetectability of the underlying steganography scheme. We now prove robustness for the secret key case; the public key version is analogous.

Fix a PPT adversary $\mathcal{A}$ and a security parameter $\lambda \in \mathbb{N}$. Let $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, let $\zeta \leftarrow \mathcal{A}^{\overline{\mathsf{Wat}}_{\mathsf{sk}}, \overline{\mathsf{Ver}}_{\mathsf{vk}}}(1^\lambda)$, and let Π denote the transcript $\mathsf{Trans}(\mathcal{A}^{\overline{\mathsf{Wat}}_{\mathsf{sk}}})$. Assume, toward contradiction, that there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that

$$\Pr\left[\begin{array}{r}(\exists \zeta^* \preceq \zeta, (x, y) \in \Pi \text{ such that } \mathsf{EBC}[\Phi, n](\zeta^*, y) = 1 \wedge\ |\zeta^*| = n) \\ \wedge\ \overline{\mathsf{Ver}}(\mathsf{vk}, \zeta) = 0\end{array}\right] \geq 1/q(\lambda)\ .$$

Whenever the above event occurs, the predicate $\mathsf{EBC}[\Phi, n](\zeta^*, y) = 1$, with $|\zeta^*| = n$, guarantees the existence of a block index $t \in \mathbb{N}$ for which

$$\Phi(\zeta^*, y_t) = 1\ .$$

Because each block y_t is produced by $\mathsf{Embed}_{\mathsf{sk}}$ during $\mathcal{A}$'s interaction with $\overline{\mathsf{Wat}}_{\mathsf{sk}}$, we have

$$(x, y) \in \mathsf{Trans}(\mathcal{A}^{\overline{\mathsf{Wat}}_{\mathsf{sk}}}) \ \Rightarrow\ \Big((\pi = (x, y_1, \ldots, y_{t-1}),\ m = 1),\ y_t\Big) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Embed}_{\mathsf{sk}}})\ .$$

Yet in the same event, we also have $\overline{\mathsf{Ver}}(\mathsf{vk}, \zeta) = 0$, which implies $\mathsf{Dec}(\mathsf{vk}, \zeta^*) = \perp$, since failed verification guarantees that *every* n-bit substring of ζ decodes to $\perp$. Consequently, there exists an adversary $\mathcal{B}$ that simulates $\mathcal{A}$ and outputs the substring ζ^*, which is Φ-close to an embedding block y_t but fails to decode with probability at least $1/q(\lambda)$. This violates the Φ-block robustness of the underlying steganography scheme, yielding a contradiction. □

Block steganography instantiations. We present existing constructions for both secret and public key robust block steganography schemes for language models with *sufficiently high entropy*.

Secret key block steganography from PRCs. For the secret key setting, recall PRCs, which closely resemble block steganography schemes for the uniform language model (e.g., $Q(\pi) = 1/2$ for all $\pi \in \{0,1\}^*$). Christ and Gunn [CG24] showed that PRCs can be used to build a robust watermarking scheme. However, their construction leverages standard PRCs, which achieve a weaker form of robustness, where the robustness adversary is modeled as an error channel acting independently of the encoding and decoding keys. By plugging *ideal* PRCs [AACD+24] (as in Definition 4.4) into their construction, we essentially obtain the adaptive guarantee that we require.

A subtlety is that ideal PRCs guarantee that an input ξ decodes to a *random* message m whose encoding y is close to ξ, rather than the *unique* message m as required in block steganography. We show that this distinction is harmless: due to pseudorandomness, no efficient adversary can find any ξ that is simultaneously close to the encodings of two distinct messages.

Theorem 4.14 (Secret-key block steganography from ideal PRC [CG24; AACD+24])**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, let $n = \mathrm{poly}(\lambda)$ with $n \geq \lambda$ be any block size, and let Q be any language model with min-entropy rate at least a constant. That is, there exists a constant $\epsilon \in (0, \frac{1}{4})$ such that*

$$\forall \pi \in \{0,1\}^*, y \leftarrow \overline{Q}_\ell(\pi) \text{ satisfies } H_\infty(y) \geq 4\Big(\sqrt{\frac{1}{2} - \epsilon}\Big)n$$

If there exists an ideal pseudorandom code with constant information rate $\rho \in (0,1)$ and constant error rate $\delta \in (0, \frac{1}{4})$, then there exists a block size n secret key $\mathsf{Ham}_{\delta-\epsilon}$*-robust steganography scheme for Q with message size $k = \rho n$.*

Proof. Let $\mathsf{PRC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a block size n ideal PRC, assumed to exist in the theorem. We define $\mathsf{Steg} = (\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$, which mirrors the watermarking scheme of [CG24] but allows embedding messages and only outputs n-sized blocks.

- $\mathsf{Steg.Gen}(1^\lambda)$: samples encoding and decoding keys $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PRC.Gen}(1^\lambda)$ and outputs $(\mathsf{vk} = \mathsf{dk}, \mathsf{sk} = \mathsf{ek})$.
- $\mathsf{Steg.Embed}(\mathsf{sk}, \pi, m)$: encodes the message $c \leftarrow \mathsf{PRC.Enc}(\mathsf{sk}, m)$ and then samples each bit of the response block $y \in \{0,1\}^n$ as

  $$y[i] \leftarrow \mathsf{Ber}\Big(p_i - (-1)^{c[i]} \cdot \min\{p_i, 1 - p_i\}\Big), \quad \forall i \in [n],$$

  where $p_i = Q(\pi, y[1:i])$ is the probability that the i-th bit should be 1 under Q. Then outputs y.
- $\mathsf{Steg.Dec}(\mathsf{vk}, \zeta \in \{0,1\}^n)$ outputs $m' \leftarrow \mathsf{PRC.Dec}(\mathsf{vk}, \zeta)$.

Undetectability follows from pseudorandomness of PRC [CG24, Lemma 19]. For robustness fix $\lambda \in \mathbb{N}$ and a PPT adversary $\mathcal{A}$, and let $(\mathsf{PRC.ek}, \mathsf{PRC.dk}) \leftarrow \mathsf{PRC.Gen}(1^\lambda)$ and $\zeta \leftarrow \mathcal{A}^{\mathsf{Embed}_{\mathsf{sk}}, \mathsf{Dec}_{\mathsf{vk}}}(1^\lambda)$. We will show that $\mathcal{A}$ cannot break the Φ-block robustness of Steg.

We move to a hybrid in which the oracles $\mathsf{Embed}_{\mathsf{sk}}$ and $\mathsf{Dec}_{\mathsf{vk}}$ are implemented using $\mathsf{PRC.Enc}^{\mathsf{ideal}}$ and $\mathsf{PRC.Dec}^{\mathsf{ideal}}$ respectively (the oracles given in the ideal PRC game 4, $\mathcal{G}^{\mathsf{ideal}}_{\lambda,\delta}(\mathcal{A})$). By the ideal security of PRC, this hybrid is computationally indistinguishable to $\mathcal{A}$.

Now consider the event that ζ is $\mathsf{Ham}_{\delta-\epsilon}$-close to some block $y \in \{0,1\}^n$ for which

$$(\pi, m, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Steg.Embed}_{\mathsf{sk}}}) \,.$$

We will show that ζ decodes to m with overwhelming probability. Since Q has entropy rate at least a constant, every response y has high empirical entropy as in [CG24, Definition 7]. From [CG24, Lemma 21], y must therefore be Ham_ϵ-close to the PRC codeword c used to sample it with all but negligible probability. That is, with probability at least $1 - \mathrm{negl}(\lambda)$, there exists a codeword $c \in \{0,1\}^n$ such that

$$(m, c) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{PRC.Enc}^{\mathsf{ideal}}}) \text{ and } \mathsf{Ham}_\epsilon(y, c) = 1\ . \tag{1}$$

By the triangle inequality, ζ is Ham_δ-close to c. We want to argue that the ideal decoder will therefore decode ζ to the corresponding message m. However, the ideal decoder only guarantees decoding to *a random* message satisfying 1. We now show that m is, with overwhelming probability, the unique such message, which will conclude the argument.

Since $\mathsf{PRC.Enc}^{\mathsf{ideal}}$ generates a fresh random codeword for each input, all codewords in the transcript are, with overwhelming probability, pairwise at a Hamming distance greater than $2\delta n$. Formally, let

$$C \ = \ \{\, c : \exists m' \text{ with } (m', c) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Enc}^{\mathsf{ideal}}}) \,\}.$$

By Lemma 4.15 and the fact that codewords from the ideal encoder are uniformly random strings, the probability that any two distinct $c, c' \in C$ satisfy

$$\mathsf{Ham}_{2\delta}(c, c') = 1$$

is negligible in $n \geq \lambda$. Since $|C|$ is polynomial in λ, a union bound over all pairs implies that, with negligible probability, no pair of distinct codewords in C are $\mathsf{Ham}_{2\delta}$-close. Consequently, if ζ is Ham_δ-close to some codeword $c \in C$, then it cannot be Ham_δ-close to any other codeword in C.

Combining this fact with 1, we have that whenever ζ is $\mathsf{Ham}_{\delta-\epsilon}$-close to some block y encoding a message m, the ideal decoder outputs m with all but negligible probability. Since the ideal decoder is indistinguishable from $\mathsf{PRC.Dec}_{\mathsf{vk}}$ to $\mathcal{A}$, it follows that Steg satisfies $\mathsf{Ham}_{\delta-\epsilon}$-block robustness. □

The proof above relied on a simple fact, that random strings are far in Hamming distance with overwhelming probability.

Lemma 4.15. *Let $n \in \mathbb{N}$ be any string length and let $\delta \in (0, \frac{1}{2})$ be a fixed constant. Then*

$$\Pr_{c,c' \leftarrow \{0,1\}^n} \left[\mathsf{Ham}_\delta(c, c') = 1\right] \leq \mathrm{negl}(n).$$

Proof. For independently and uniformly random $c, c' \leftarrow \{0,1\}^n$, each coordinate differs with probability $1/2$, so

$$\mathbb{E}[\|c - c'\|_0] = \frac{n}{2}.$$

Applying a Chernoff bound with $t = 1 - 2\delta \in (0, 1)$, we get

$$\Pr\left[\|c - c'\|_0 \leq \delta n\right] \leq \exp\left(-\frac{n(1-2\delta)^2}{4}\right).$$

Since δ is a constant strictly less than $1/2$, the exponent is a negative constant times n, and therefore the probability is negligible in n. □

Public key block steganography in random oracle model. Fairoze et al. [FGJM+25] leverage random oracles to build a *pubic key* watermarking scheme. We observe that their scheme is implicitly built on top of a public key block steganography scheme, which we formalize in the following theorem.

Theorem 4.16 (Public key block steganography from random oracles [FGJM+25])**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $n = \mathrm{poly}(\lambda)$ be any block size, and Q be any language model satisfying the following entropy condition: for some $\ell = \mathrm{poly}(\lambda)$, size ℓ outputs from Q have high min-entropy. That is,*

$$\forall \pi \in \{0,1\}^*, y \leftarrow \overline{Q}_\ell(\pi) \text{ satisfies } H_\infty(y) \geq \omega(\log \lambda) \ .$$

For simplicity, assume ℓ divides n, so that n/ℓ is an integer.

Then there exists constants $\delta \in (0, \frac{1}{2})$, $\rho \in (0,1)$ where in the random oracle model, there exists a block size n public key *Φ_δ-robust block steganography scheme with message size $k = \rho(n/\ell - 1)$, where $\Phi_\delta : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is the predicate*

$$\Phi_\delta(y, y') : \begin{pmatrix} \text{Parse } (y_1, \ldots, y_{n/\ell}) = \mathsf{Blocks}(y),\ (y'_1, \ldots, y'_{n/\ell}) = \mathsf{Blocks}(y') \text{ into } \ell\text{-bit blocks}, \\ \text{Output 1 iff } y_1 = y'_1 \text{ and for a } (1-\delta)\text{-fraction of } i \in \{2, \ldots, n/\ell\},\ y_i = y'_i \end{pmatrix}.$$

Proof. We adapt the construction of [FGJM+25]. Let $(\mathsf{Enc}, \mathsf{Dec})$ be an information-theoretic error-correcting code with constant rate $\rho \in (0,1)$, that can correct up to a $\delta \in (0, \frac{1}{2})$ fraction of errors[11]. Let H_1 and H_2 be independent random oracles. We define a block steganography scheme $\mathsf{Steg} = (\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec})$:

- $\mathsf{Steg.Gen}(1^\lambda)$: samples $r \leftarrow \{0,1\}^\lambda$ and outputs $\mathsf{vk} = \mathsf{sk} = r$.
- $\mathsf{Steg.Embed}(r, \pi, m)$: first samples a block of ℓ bits $y_1 \leftarrow \overline{Q}_\ell(\pi)$, then computes the masked codeword
$$c = \Big(\mathsf{Enc}(m) \oplus H_2\big((r, y_1)\big)\Big) \in \{0,1\}^{n/\ell - 1} \ .$$
The algorithm then embeds c into the next $n/\ell - 1$ blocks of size ℓ by rejection-sampling. That is,
$$y_i \leftarrow \overline{Q}_\ell((\pi, y_1, \ldots, y_{i-1})) \text{ conditioned on } H_1\big((r, y_i)\big) = c[i-1] \ ,$$
for $i \in \{2, \ldots, n/\ell\}$. Then outputs $y = (y_1, \ldots, y_{n/\ell}) \in \{0,1\}^n$.
- $\mathsf{Steg.Ver}(r, \zeta \in \{0,1\}^n)$: parses $\zeta = (\zeta_1, \ldots, \zeta_{n/\ell})$ into blocks of size ℓ, then computes
$$\mathsf{pad} = H_2\big((r, \zeta_1)\big), \text{ and } c'[i-1] = H_1\big((r, \zeta_i)\big) \text{ for } i \in \{2, \ldots, n/\ell\},$$
and outputs $m = \mathsf{Dec}(c' \oplus \mathsf{pad})$.

Undetectability and the fact that the rejection sampling succeeds for every block with all but negligible probability follows directly from [FGJM+25, Lemma 3.2]. Note that using their proof requires masking the message with the random oracle value of the first block $H_2\big((r, y_1)\big)$.

To see robustness, let $\zeta = (\zeta_1, \ldots, \zeta_{n/\ell}) \leftarrow \mathcal{A}^{\mathsf{Embed}_r}(r)$, and suppose there exists $(\pi, m, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Embed}_r})$ with

$$\Phi_\delta(\zeta, y) = 1.$$

Denote $(y_1, \ldots, y_{n/\ell})$ as the ℓ-bit blocks of y. By definition of Φ_δ, we have:

[11] Such codes exists, for example [ABNN+92; NN90; Ta-17].

1. $y_1 = \zeta_1$, so the verifier $\mathsf{Ver}(r, \zeta)$ recovers the pad, $\mathsf{pad} = H_2((r, y_1))$, exactly.
2. Among the blocks $(\zeta_2, \ldots, \zeta_{n/\ell})$, at most a δ-fraction differ from the corresponding $(y_2, \ldots, y_{n/\ell})$. Thus at most a δ-fraction of the bits of c' computed by $\mathsf{Steg.Ver}$ differ from the initial encoding $c = \mathsf{Enc}(m) \oplus \mathsf{pad}$.

Because the error-correcting code can correct up to a δ-fraction of errors, $\mathsf{Dec}(c' \oplus \mathsf{pad}) = m$. $\square$

5 Robust and Recoverable Digital Signatures

We present our formal definition of robust digital signatures, which allow verification even in the presence of small, adversarial perturbations to the message.

Definition 5.1 (Robust digital signatures with strong unforgeability)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter and let Φ be any predicate. A digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is robust and strongly unforgeable with respect to Φ if for any PPT adversary $\mathcal{A}$, the scheme satisfies robustness,*

$$\Pr\left[\begin{array}{rcl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ (\zeta, \sigma) & \leftarrow & \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk}) \end{array} : \begin{array}{r} \Big(\exists (y, \sigma) \in \Pi^{12} \text{ such that } \Phi(\zeta, y) = 1\Big) \\ \wedge\ \mathsf{Ver}(\mathsf{pk}, \zeta, \sigma) = 0 \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

and strong unforgeability,

$$\Pr\left[\begin{array}{rcl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ (\zeta, \sigma) & \leftarrow & \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk}) \end{array} : \begin{array}{r} \mathsf{Ver}(\mathsf{pk}, \zeta, \sigma) = 1 \\ \wedge\ \Big(\forall (y, \sigma) \in \Pi,\ \Phi(\zeta, y) = 0\Big) \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

where Π is the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}})$.

A recoverable digital signature scheme, in turn, requires a PPT recovery algorithm Rec that can recover the original message from its signature and a nearby, adversarially chosen input. For a digital signature scheme with message size $n = n(\lambda)$, signature size $k = k(\lambda)$, Rec has the syntax

- $\mathsf{Rec}(\mathsf{pk}, \zeta, \sigma)$: takes in the public key pk, a string $\zeta \in \{0,1\}^n$, and a signature $\sigma \in \{0,1\}^k$ and outputs a recovered message or a failure symbol $y_{\mathsf{rec}} \in \{0,1\}^n \cup \{\perp\}$.

For Rec to satisfy strong unforgeability, any recovered message y_{rec} must indeed be close to the input ζ and must have been originally signed with the input signature σ.

Definition 5.2 (Recoverable digital signatures)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter and let Φ be any predicate. A digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Rec})$ is recoverable with respect to Φ if for any PPT adversary $\mathcal{A}$, the scheme is robustly recoverable, meaning*

$$\Pr\left[\begin{array}{rcl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ (\zeta, \sigma) & \leftarrow & \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk}) \\ y_{\mathsf{rec}} & \leftarrow & \mathsf{Rec}(\mathsf{pk}, \zeta, \sigma) \end{array} : \begin{array}{r} \Big(\exists (y, \sigma) \in \Pi \text{ such that } \Phi(\zeta, y) = 1\Big) \\ \wedge\ y_{\mathsf{rec}} \neq y \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

and unforgeably recoverable, meaning

$$\Pr\left[\begin{array}{rcl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ (\zeta, \sigma) & \leftarrow & \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk}) \\ y_{\mathsf{rec}} & \leftarrow & \mathsf{Rec}(\mathsf{pk}, \zeta, \sigma) \end{array} : \begin{array}{r} y_{\mathsf{rec}} \neq \perp \\ \wedge\ \Big((y_{\mathsf{rec}}, \sigma) \notin \Pi\ \vee\ \Phi(\zeta, y_{\mathsf{rec}}) = 0\Big) \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

where Π is the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}})$.

[12] We slightly overload notation by quantifying over tuples when some components are fixed by context. For example here, the statement $\exists (y, \sigma) \in \Pi$, where σ is the signature output by the adversary, means $\exists y$ such that $(y, \sigma) \in \Pi$.

Another property we introduce is a notion of *uniqueness* for signatures. While this property is not required for our main result, it will be useful to guarantee a certain chaining property that we present in Section 6.2. A digital signature has unique signatures if no efficient adversary can produce an output ζ and two messages (y, y') with the same signature such that ζ is close to one message and far from another. Intuitively, this prevents an adversary from generating duplicate signatures on messages for which it can produce an intermediate output. Note that this property can be easily achieved by appending $\omega(\log \lambda)$ random bits to each signature, to ensure *all* signatures are almost surely unique. However, we show that our construction inherently satisfies it.

Definition 5.3 (Unique signatures)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter and let Φ be any predicate. A digital signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *has unique signatures with respect to Φ, if for any PPT $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) & \\ (\zeta, y, y') \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk}) & : \begin{array}{r} \Big(\Phi(\zeta, y) = 1 \ \wedge \ \Phi(\zeta, y') = 0\Big) \\ \wedge \Big(\exists \sigma \text{ such that } (y,\sigma) \in \Pi \text{ and } (y',\sigma) \in \Pi\Big) \end{array} \end{array}\right] \leq \mathrm{negl}(\lambda)\ ,$$

where Π is the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}})$.

5.1 Difference-recovering hash family

The core ingredient of our signature scheme is a property-preserving hashing scheme (PPH) [BLV19]. We observe that existing Hamming PPH constructions [FLS22; HLTW22] support recovery of the bitwise difference $\Delta = x \oplus x'$ between two close strings x and x' using only their hashes $h(x)$ and $h(x')$. We define a slightly weaker property than this, which requires recovering the difference between two inputs when one of the inputs is given in the clear. This property is sufficient for watermarking applications, and we show that it can be achieved by *any* PPH for the Hamming predicate. In Section A, we define the stronger variant, which requires recovering the difference using only the hashes of both inputs.

Definition 5.4 (Difference recovering PPH)**.** *For any predicate Φ, a hash family* $(\mathsf{Sample}, \{\mathcal{H}_\lambda\})$ *is difference-recovering w.r.t. Φ if there exists a PPT algorithm* Rec *such that for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} h & \leftarrow \mathsf{Sample}(1^\lambda) \\ (x, x') & \leftarrow \mathcal{A}(1^\lambda, h) \\ \Delta & \leftarrow \mathsf{Rec}(h, h(x), x') \end{array} : \begin{array}{r} (\Delta \neq x \oplus x' \wedge \Phi(x, x') = 1) \\ \vee\ (\Delta \neq \perp \wedge \Phi(x, x') = 0) \end{array}\right] \leq \mathrm{negl}(\lambda)$$

Proposition 5.5. *Let λ be the security parameter. For any input size $n = \mathrm{poly}(\lambda)$ and threshold $r = \mathrm{poly}(\lambda)$ such that $r < n/2$, every $\mathsf{Ham}_{r/n}$-PPH supports difference recovery w.r.t. $\mathsf{Ham}_{r/n}$.*

Proof. Let $h \leftarrow \mathsf{Sample}(1^\lambda)$. We construct a recovery algorithm $\mathsf{Rec}(h, \cdot, \cdot)$ which, given a hash $h(x)$ and an input x' in the clear, returns the difference $x \oplus x'$ if $\mathsf{Ham}_{r/n}(x, x') = 1$, and returns $\perp$ otherwise. The recovery algorithm only needs oracle access to $\mathsf{Hash}(h, \cdot)$ and $\mathsf{Eval}(h, h(x), \cdot)$. For the reader's convenience, we include the pseudocode for Rec in Figure 3.

We may assume that $\mathsf{Eval}(h, h(x), h(\zeta)) = \mathsf{Ham}_{r/n}(x, \zeta)$ for *all* inputs $\zeta \in \mathbb{F}_2^n$ used by Rec to query $\mathsf{Eval}(h, h(x), \cdot)$, since Rec is an efficient deterministic algorithm. Any violation would contradict the direct-access robustness of the underlying Hamming PPH.

If $\mathsf{Eval}(h, h(x), h(x')) = 0$, then $\mathsf{Rec}(h, h(x), x')$ returns $\perp$, which is fine since $\mathsf{Ham}_{r/n}(x, x') = \mathsf{Eval}(h, h(x), h(x')) = 0$.

Algorithm 3 Rec

input: (h, z, x')
if If $\mathsf{Eval}(h, z, h(x')) = 0$ **then**
  return $\perp$
end if
$\Delta \leftarrow 0^n$
$\zeta \leftarrow \mathsf{GetBoundaryPoint}(h, z, x')$
for $j \in [n]$ **do**
  if $\mathsf{Eval}(h, z, h(\zeta \oplus e_j)) = 1$ **then**
    $\Delta \leftarrow \Delta \oplus e_j$
  end if
end for
return $\Delta \oplus x' \oplus \zeta$

Algorithm 4 GetBoundaryPoint

input: (h, z, x')
$\zeta \leftarrow x'$
for $j \in [n]$ **do**
  $\zeta \leftarrow \zeta \oplus e_j$
  if $\mathsf{Eval}(h, z, h(\zeta)) = 0$ **then**
    return $\zeta \oplus e_j$
  end if
end for
return $\perp$

Figure 3: Generic PPH recovery algorithm

If $\mathsf{Eval}(h, h(x), h(x')) = 1$, then $\mathsf{Rec}(h, h(x), x')$ needs to return the difference $x \oplus x'$. To this end, it first finds a *boundary point* relative to x. That is, a point $\zeta \in \mathbb{F}_2^n$ such that $\|x \oplus \zeta\|_0 = r$. Once such a point is found, difference recovery is straightforward: since we know that $\zeta \oplus x$ is r-sparse, we can determine its support of the by flipping each bit of ζ individually. Specifically, for each $j \in [n]$, we test whether $\mathsf{Eval}(h, h(x), h(\zeta \oplus e_j)) = 1$, where $e_j \in \mathbb{F}_2^n$ denotes the j-th standard basis vector (i.e., the vector with a 1 in the j-th position and zeros elsewhere). We conclude that j lies in the support of $x \oplus \zeta$ if and only if the evaluation remains 1 after the one-bit perturbation.

We now describe an efficient procedure for finding a boundary point ζ with respect to x, given $h(x)$ and x' satisfying $\|x \oplus x'\|_0 \leq r$. For any $j \in [n]$, define

$$u_j = e_1 \oplus \cdots \oplus e_j \ , \quad \zeta_j = x' \oplus u_j \ .$$

Define the potential function $f : \{0, 1, \ldots, n\} \to \{0, 1, \ldots, n\}$ by

$$f(j) = \|x \oplus \zeta_j\|_0 \ ,$$

with $f(0) = \|x \oplus x'\|_0$.

Observe that for each $j < n$, we have $|f(j+1) - f(j)| = 1$. In other words, the value of $f(j)$ always changes by either 1 or -1. In addition, it holds that $f(0) \leq r$ and $f(n) = n - f(0) > r$. Thus, by the discrete intermediate value property, there exists $s \in \{0, 1, \ldots, n\}$ such that $f(s) = r$. The corresponding ζ_s is the desired boundary point.

Although Rec cannot explicitly compute $f(j)$, it can determine whether $f(j) \leq r$ by querying $\mathsf{Eval}(h, h(x), h(\zeta_j))$. The algorithm increments j from 1 to n, stopping at the first index where Eval returns 0. Let j be the smallest such index. Then the boundary point is ζ_{j-1}. □

Difference recovery from standard CRHF. Proposition 5.5 implies that the PPH from homomorphic CRHFs given in Theorem 4.7 satisfies difference recovery with respect to Ham_δ. However, the converse does not hold. In particular, one can instantiate a difference recovering hash family for Hamming that does not admit an evaluation algorithm Eval for computing the predicate on two hashes, as the recovery algorithm Rec receives one of its inputs in the clear. This instantiation is heavily inspired by the notion and construction of *secure sketches* [DORS08]. We refer the reader to Section B for additional details.

Theorem 5.6 (Sharp sketch for Hamming, adapted from [DORS08, Section 8.3])**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and let $n = \mathrm{poly}(\lambda)$ denote the input size. Suppose there exists a CRHF with output size $\ell = \ell(\lambda)$. For any constants $\delta \in (0, 1/2)$ and $\rho > H_2^{\mathsf{BZ}}(\delta)$, where $H_2^{\mathsf{BZ}}(\cdot)$ refers to the Blokh-Zyablov bound, there exists a difference recovering hash family* $(\mathsf{Sample}, \mathsf{Rec})$ *w.r.t.* Ham_δ *that has hash size $\rho \cdot n + \ell$.*

Remark 5.7 (Comparison with Theorem 4.7)**.** *The hash family from Theorem 5.6 requires only a standard collision-resistant hash function, not a homomorphic one. Moreover, it uses efficiently list-decodable codes over $\mathbb{F}_2$, whereas Theorem 4.7 uses codes over $\mathbb{F}_3$. As a result, it benefits from the Blokh-Zyablov bound over $\mathbb{F}_2$ and avoids the multiplicative factor of $\log_2 3$ in the hash size. However, it is not a PPH.*

5.2 Robust signature from PPH

We now prove a generic theorem showing that the construction presented in Section 2.1 is both robust and unforgeable with respect to Φ. In Section 5.4, we instantiate the theorem with an existing Hamming PPH [HLTW22]. We recall the construction in Figure 4.

Algorithm 5 RGen

input: (1^λ)
$h \leftarrow \mathsf{Sample}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
return $((h, \mathsf{pk}), \mathsf{sk})$

Algorithm 6 RSign

input: $((h, \mathsf{pk}), \mathsf{sk}, y)$
$z \leftarrow h(y)$
$\tau \leftarrow \mathsf{Sign}(\mathsf{sk}, z)$
return (z, τ)

Algorithm 7 RVer

input: $((h, \mathsf{pk}), \zeta, \sigma)$
$(z, \tau) \leftarrow \sigma$
if $\mathsf{Ver}(\mathsf{pk}, z, \tau) = 1$ **then**
  $b \leftarrow \mathsf{Eval}(h, h(\zeta), z)$
  return b
end if
return 0

Figure 4: Robust digital signature construction

Theorem 5.8 (Generic robust signatures)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, $n = \mathrm{poly}(\lambda)$ be any message size, and Φ be any predicate. If* $\mathsf{PPH}[\mathsf{Sample}, \mathsf{Hash}, \mathsf{Eval}]$ *is a Φ-PPH with hash size $k = \mathrm{poly}(\lambda)$ and* $\mathsf{DSS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ *is a strongly unforgeable digital signature scheme with signature size $\ell = \mathrm{poly}(\lambda)$, then* $(\mathsf{RGen}, \mathsf{RSign}, \mathsf{RVer})$*, defined in Figure 4, is a robust and unforgeable signature scheme for Φ with signature size $k + \ell$. Moreover, the scheme has unique signatures with respect to Φ.*

Proof. Robustness of RDS follows immediately from the correctness of the underlying DSS and the direct-access robustness of PPH. We focus on proving strong unforgeability. Assume for contradiction that there exists a PPT adversary $\mathcal{A}$ that outputs a pair (ζ, σ) such that $\Phi(\zeta, y) = 0$ for all $(y, \sigma) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RSign}})$ but the pair passes RVer with non-negligible probability.

That is, let $(\mathsf{RDS.pk}, \mathsf{RDS.sk}) \leftarrow \mathsf{RGen}(1^\lambda)$, $(\zeta, \sigma) \leftarrow \mathcal{A}^{\mathsf{RSign}_{\mathsf{RDS.sk}}}(\mathsf{RDS.pk})$, and let Π denote the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{RSign}})$, where each entry logs a tuple (y, z, τ), with $z = h(y)$ and $\tau = \mathsf{DSS.Sign}(\mathsf{DSS.pk}, z)$. Then there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that

$$\Pr\left[\begin{array}{r} \mathsf{RVer}(\mathsf{RDS.pk}, \zeta, \sigma) = 1 \\ \wedge\, \forall (y, z, \tau) \in \Pi, \Phi(\zeta, y) = 0 \end{array}\right] > 1/q(\lambda) \,.$$

Let E_{DSS} denote the event that $\sigma = (z, \tau)$ constitutes a successful forgery against DSS , and let E_{PPH} denote the event that (ζ, z) violates the direct-access robustness of PPH. More precisely,

$$E_{\mathsf{DSS}} = \mathsf{DSS.Ver}(\mathsf{pk}, z, \tau) = 1 \ \wedge \ (\cdot, z, \tau) \notin \Pi \ ,$$
$$E_{\mathsf{PPH}} = \mathsf{Eval}(h, h(\zeta), z) = 1 \ \wedge \bigvee_{(y,z,\tau)\in\Pi} \Phi(\zeta, y) = 0 \ ,$$

where $(\cdot, z, \tau) \notin \Pi$ is shorthand for $\nexists y$ such that $(y, z, \tau) \in \Pi$. Since

$$\mathsf{RVer}(\mathsf{RDS.pk}, \zeta, \sigma) = \mathsf{DSS.Ver}(\mathsf{DSS.pk}, z, \tau) \wedge \mathsf{Eval}(h, h(\zeta), z) \ ,$$

we observe that

$$\mathsf{RVer}(\mathsf{RDS.pk}, \zeta, \sigma) = 1 \ \wedge \ (\forall (y, \sigma) \in \Pi, \Phi(\zeta, y) = 0) \subseteq E_{\mathsf{DSS}} \vee (E_{\mathsf{PPH}} \wedge (\cdot, z, \tau) \in \Pi) \ .$$

It follows that $\Pr\left[E_{\mathsf{DSS}} \vee (E_{\mathsf{PPH}} \wedge (\cdot, z, \tau) \in \Pi)\right] > 1/q(\lambda)$. This leads to a contradiction, since both E_{DSS} and $E_{\mathsf{PPH}} \wedge (\cdot, z, \tau) \in \Pi$ should occur with negligible probability due to the the strong unforgeability of DSS, and the direct-access robustness of PPH respectively. Therefore, the robust signature scheme for Φ presented in Figure 4 is strongly unforgeable.

Unique signatures. Suppose, for the sake of contradiction, that there exists an adversary $\mathcal{A}$ and a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RGen}(1^\lambda)$ and $(\zeta, y, y') \leftarrow \mathcal{A}^{\mathsf{RSign}_{\mathsf{sk}}}(\mathsf{pk})$,

$$\Pr\left[\begin{array}{r} \exists \sigma \text{ such that } (y, \sigma) \in \Pi \text{ and } (y', \sigma) \in \Pi \\ \wedge \ \Phi(\zeta, y) = 1 \ \wedge \ \Phi(\zeta, y') = 0 \end{array}\right] > 1/q(\lambda) \ ,$$

Then $\mathcal{A}$ finds y and y' such that $h(y) = h(y')$, where h is the PPH hash function, with probability at least $1/q(\lambda)$, because their signatures are identical. However, this implies that either the pair (ζ, y) or (ζ, y') violates the robustness of PPH. Specifically, the evaluation algorithm Eval cannot distinguish $h(y)$ from $h(y')$, and hence $\mathsf{PPH.Eval}(h, h(y), h(\zeta))$ and $\mathsf{PPH.Eval}(h, h(y'), h(\zeta))$ will have the same output distribution, even though $\Phi(\zeta, y) \neq \Phi(\zeta, y')$. Therefore, at least one of these pairs must cause Eval to output the incorrect predicate value with inverse-polynomial probability, contradicting the robustness of PPH. □

5.3 Recoverable signatures from difference-recovering PPH

We show that our generic construction from Theorem 5.8 is recoverable, provided the underlying PPH is difference recovering. Let $(\mathsf{RGen}, \mathsf{RSign}, \mathsf{Rec})$ denote the scheme from Figure 4, where the verification algorithm RVer is replaced by the recovery algorithm Rec presented in Algorithm 8. The guarantees of this scheme are formally stated in Corollary 5.9.

Algorithm 8 Rec

1: **input**: $(\mathsf{pk} = (h, \mathsf{DSS.pk}), \zeta, \sigma' = (z, \tau))$
2: **if** $\mathsf{RVer}(\mathsf{pk}, \zeta, \sigma) = 0$ **then**
3: return $\perp$
4: **end if**
5: $\Delta \leftarrow \mathsf{PPH.Rec}(h, z, \zeta)$
6: return $\zeta \oplus \Delta$

Corollary 5.9 (Recoverable signatures)**.** *Let Φ be any predicate. If* $\mathsf{PPH}[\mathsf{Sample}, \mathsf{Hash}, \mathsf{Rec}]$ *is difference recovering w.r.t. Φ (as in Definition 5.4) and* $\mathsf{DSS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ *is a strongly unforgeable digital signature scheme, then the resulting* $(\mathsf{RGen}, \mathsf{RSign}, \mathsf{Rec})$ *is a recoverable digital signature scheme w.r.t. Φ.*

Remark 5.10 (Recoverable signatures without PPH)**.** *Note that Corollary 5.9 can be instantiated using* any *difference recovering hash family, not necessarily a PPH. Although* RVer*, invoked by* Rec*, leverages PPH evaluation* $\mathsf{Eval}(h, h(\zeta), z)$*, this call can instead be replaced by computing* $\Phi(\zeta, \mathsf{PPH.Rec}(h, z, \zeta) \oplus \zeta)$*. We present the construction using* Eval *only because PPHs are widely studied.*

Proof. Let $(\mathsf{pk} = (h, \mathsf{DSS.pk}), \mathsf{sk}) \leftarrow \mathsf{RGen}(1^\lambda)$ and let $\zeta \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}}(\mathsf{pk})$ be the adversary's output.

Robust recovery. Suppose there exists y such that $(y, \sigma) \in \Pi$ and $\Phi(\zeta, y) = 1$. By robustness of RDS, $\mathsf{RVer}(\mathsf{pk}, \zeta, \sigma) = 1$ with probability at least $1 - \mathrm{negl}(\lambda)$. Conditioned on this, $\sigma = (h(y), \cdot)$, and the difference-recovery of PPH satisfies $\mathsf{PPH.Rec}(h, h(y), \zeta) = y \oplus \zeta$ with probability at least $1 - \mathrm{negl}(\lambda)$, so the final output is y with overwhelming probability.

Unforgeable recovery. Suppose $\mathsf{Rec}(\mathsf{pk}, \zeta, \sigma) = y_{\mathsf{rec}}$, with $y_{\mathsf{rec}} \neq \perp$. Then by definition of Rec, $\mathsf{RVer}(\mathsf{pk}, \zeta, \sigma) = 1$. By strong unforgeability of RDS, with probability at least $1 - \mathrm{negl}(\lambda)$ there exists y such that $(y, \sigma) \in \Pi$ and $\Phi(\zeta, y) = 1$, meaning $\sigma = (h(y), \cdot)$ corresponds to a valid signature of y.

Conditioned on the above, difference-recovery of PPH yields $\mathsf{PPH.Rec}(h, h(y), \zeta) = y \oplus \zeta$ with probability at least $1 - \mathrm{negl}(\lambda)$, meaning

$$y_{\mathsf{rec}} = \zeta \oplus \mathsf{PPH.Rec}(h, h(y), \zeta) = \zeta \oplus (y \oplus \zeta) = y,$$

which implies that y_{rec} satisfies $(y_{\mathsf{rec}}, \sigma) \in \Pi$ and $\Phi(\zeta, y_{\mathsf{rec}}) = 1$. Since the intersection of both events occurs with overwhelming probability, the claim follows. □

5.4 Instantiating Hamming signatures

We instantiate a recoverable digital signature scheme for Hamming predicates, providing concrete trade-offs between the signature size k (relative to the message length n) and the tolerable Hamming error r. Since the regime of practical interest in watermarking corresponds to a constant error rate $r/n = \Omega(1)$, we focus on the rates achievable in this setting.

Here, we refer to the signature-to-message rate, k/n of a RDS with signature size k, message size n.

Corollary 5.11 (Recoverable digital signatures with constant error-rate)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and suppose there exists a CRHF with output size $\ell = \ell(\lambda)$ and a strongly unforgeable digital signature scheme with signature size $k = k(\lambda)$. Let $n = \mathrm{poly}(\lambda)$ such that $(\ell + k)/n = o(1)$ be the message size. For any constants $\delta \in (0, 1/2)$ and $\rho > H_2^{\mathsf{BZ}}(\delta)$, where $H_2^{\mathsf{BZ}}(\cdot)$ denotes the Blokh–Zyablov bound, there exists a recoverable digital signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Rec})$ *for the predicate* Ham_δ *over $\{0,1\}^n$ with signature-to-message rate $\rho + o(1)$. Moreover, the scheme has unique signatures with respect to* Ham_δ*.*

Proof. We instantiate Corollary 5.9 using the difference recovering hash family from Theorem 5.6 for the predicate Ham_δ. The hash size is $\rho n + \ell$, and signing the hash incurs an additional k bits. Thus, the total signature size is $\rho n + \ell + k$, and the resulting recoverable signature-to-message rate is $(\rho n + \ell + k)/n = \rho + o(1)$. □

Remark 5.12. *We instantiate our construction using a sharp sketch (Theorem 5.6) rather than a PPH. Note that we could alternatively use the PPH from Theorem 4.7, with recoverability following from Proposition 5.5. However, using a secure sketch offers the advantage that we can achieve any constant $\rho > H_2^{\mathsf{BZ}}(\delta)$ (rather than $\rho > H_3^{\mathsf{BZ}}(\delta)\log_2 3$), and that we can rely only on CRHFs instead of homomorphic CRHFs.*

We can also instantiate signatures that achieve a subconstant error rate, by using a sharp sketch and analysis along the lines of [HLTW22, Corollary 6.8].

Corollary 5.13 (Recoverable digital signatures with subconstant error-rate)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and suppose there exists a CRHF with output size $\ell = \ell(\lambda)$ and a strongly unforgeable digital signature scheme with signature size $k = k(\lambda)$. Let $n = \mathrm{poly}(\lambda)$ be the message size. For any $\delta = o(1)$, there exists a recoverable digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Rec})$ for the predicate Ham_δ with signature size $O(\delta n \log n) + \ell + k$. Moreover, the scheme has unique signatures with respect to Ham_δ.*

6 Recoverable, Robust, and Unforgeable Watermarking

In Section 6.1, we present our main construction, which combines block steganography with robust signatures to obtain a robust and unforgeable watermarking scheme. We then demonstrate, in Section 6.2, that our verification algorithm can be generalized by chaining across longer substrings, yielding fine-grained control over the tradeoff in substring size between robustness and unforgeability. In Section 6.3, we prove that if the underlying signature scheme is recoverable, we can build a *recoverable* watermarking scheme. Finally, in Section 6.4, we instantiate both secret and public key versions of our construction using existing block steganography constructions [CG24; AACD+24; FGJM+25] together with our robust and recoverable digital signature instantiations (Corollaries 5.11 and 5.13).

6.1 Robust and unforgeable watermarking

Theorem 6.1 summarizes the guarantees of our watermarking scheme. It has the following properties

- $(\mathsf{EBC}[\Phi, n], 2n)$-robustness: an (adversarially chosen) ζ verifies if it contains a substring $\zeta^* \preceq \zeta$ of size $2n$ whose two blocks (ζ_1^*, ζ_2^*) are Φ-close to two consecutive response blocks (y_t, y_{t+1}).
- $(\mathsf{EBC}[\Phi, n], n)$-unforgeability: whenever ζ verifies, there exists an n-sized substring $\zeta^* \preceq \zeta$ that is Φ-close to some response block y_t.

The parameters Φ and n are inherited from the underlying block steganography and robust signature schemes. Whether robustness is secret key or public key depends on the block steganography scheme. In contrast, unforgeability is inherited entirely from the robust signature and is therefore always public key. Compared to the basic watermarking scheme obtained directly from block steganography (Fact 4.13), our construction yields slightly weaker robustness—requiring a $2n$-sized close substring to verify rather than an n-sized substring—but additionally provides unforgeability (and, as we show in Section 6.3, recoverability).

Theorem 6.1 (Robust and unforgeable 2-block watermarking)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and let $k, n = \mathrm{poly}(\lambda)$ be any message and block size, respectively. Let Φ be any closeness predicate, and Q be any language model. Suppose that:*

- $\mathsf{Steg}[\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec}]$ *is a message size k, block size n block steganography scheme for Q with (secret or public key) Φ-robustness (Definition 4.11).*
- $\mathsf{RDS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ *is a message size n, signature size k Φ-robust digital signature scheme (Definition 5.1).*

Then the construction $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$ *in Figure 5 is a watermarking scheme for Q with (secret or public key)* $(\mathsf{EBC}[\Phi, n], 2n)$*-robustness, and public key* $(\mathsf{EBC}[\Phi, n], n)$*-unforgeability, where* $\mathsf{EBC}[\Phi, n]$ *is defined as in Definition 3.2.*

We present the full details of our watermarking scheme $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$, then prove the theorem. The construction is the same whether Steg is public or secret key, but its guarantees change. The Gen algorithm generates the watermarking and verification keys.

Algorithm 9 Gen

input: (1^λ)
$\mathsf{RDS.pk}, \mathsf{RDS.sk} \leftarrow \mathsf{RDS.Gen}(1^\lambda)$
$\mathsf{Steg.vk}, \mathsf{Steg.sk} \leftarrow \mathsf{Steg.Gen}(1^\lambda)$
$\mathsf{vk} = (\mathsf{RDS.pk}, \mathsf{Steg.vk}),\ \mathsf{sk} = (\mathsf{RDS.sk}, \mathsf{Steg.sk})$
return $(\mathsf{vk}, \mathsf{sk})$

Given a prompt $x \in \{0,1\}^*$, the watermarking algorithm Wat generates a response by sampling text blocks of size $n = n(\lambda)$ sequentially and embedding the robust signature associated with each block into the next.

Algorithm 10 Wat

input: $(1^\lambda, \mathsf{sk}, x)$
$k \leftarrow k(\lambda)$ `// compute signature size k`
$\sigma_1 \leftarrow \{0,1\}^k$
$y_1 \leftarrow \mathsf{Steg.Embed}(\mathsf{Steg.sk}, x, \sigma_1)$ `// sample initial block with random message`
$t \leftarrow 1$
while $\mathsf{done} \not\preceq y_t$ **do**
  $t \leftarrow t + 1$
  $\sigma_{t-1} \leftarrow \mathsf{RDS.Sign}(\mathsf{RDS.sk}, y_{t-1})$ `// generate signature of previous block`
  $y_t \leftarrow \mathsf{Steg.Embed}(\mathsf{Steg.sk}, (x, y_1, \ldots, y_{t-1}), \sigma_{t-1})$ `// embed signature into` y_t
end while
return $(y_1, \ldots, y_{t-1}, y_t[< \mathsf{done}])$ `// return the generated` y `up to the done token`

The verification algorithm Ver operates on arbitrary sized strings, by using Steg to recover embedded signatures and checking whether *any* $2n$-sized substring in the input corresponds to a valid message-signature pair.

Algorithm 11 Ver

input: $(1^\lambda, \mathsf{vk}, \zeta)$
$n \leftarrow n(\lambda)$ `// compute block size n`
for $i \in [|\zeta| - 2n + 1]$ **do**
  $\sigma_i \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta[i+n : i+2n])$ `// attempt to decode the signature`
  if $\mathsf{RDS.Ver}(\mathsf{RDS.pk}, \zeta[i : i+n], \sigma_i) = 1$ **then**
    return 1 `// output 1 if the signature verifies with the first block`
  end if
end for
return 0

Figure 5: Pseudocode for watermarking scheme $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver})$.

Proof. Undetectability follows directly from the undetectability of Steg. For robustness and unforgeability, fix $\lambda \in \mathbb{N}$ and a PPT adversary $\mathcal{A}$. Let

$$(\mathsf{RDS.pk}, \mathsf{RDS.sk}) \leftarrow \mathsf{RDS.Gen}(1^\lambda), \quad (\mathsf{Steg.vk}, \mathsf{Steg.sk}) \leftarrow \mathsf{Steg.Gen}(1^\lambda) .$$

Denote $\mathsf{vk} = (\mathsf{RDS.pk}, \mathsf{Steg.vk})$ and $\mathsf{sk} = (\mathsf{RDS.sk}, \mathsf{Steg.sk})$. Let $\zeta \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$.

Robustness. We prove the public key version; the secret-key version follows by an analogous argument. Suppose there exists a substring $\zeta^* \preceq \zeta$ with $|\zeta^*| = 2n$ and a prompt-response pair $(x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$ such that ζ^* is $\mathsf{EBC}[\Phi, n]$-close to y. We will show $\mathsf{Ver}(\mathsf{vk}, \zeta) = 1$ with probability at least $1 - \mathrm{negl}(\lambda)$. Denote $(\zeta_1^*, \zeta_2^*) = \mathsf{Blocks}(\zeta^*)$. The closeness condition says that there exists a block index $t \in \mathbb{N}$ such that

$$\Phi(\zeta_1^*, y_t) = 1 \text{ and } \Phi(\zeta_2^*, y_{t+1}) = 1 .$$

Moreover, because y_{t+1} is an n-sized block returned by $\mathsf{Wat}_{\mathsf{sk}}$, and it isn't the first block y_1, it has the signature of the previous block y_t embedded in it. That is,

$$\Big((\pi = (x, y_1, \ldots, y_t), m = \sigma_t), y_{t+1} \Big) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Steg.Embed}_{\mathsf{Steg.sk}}})$$

where σ_t is the RDS signature of y_t computed during an oracle call to $\mathsf{Wat}_{\mathsf{sk}}(x)$.

The closeness of the second half $\Phi(\zeta_2^*, y_{t+1}) = 1$, together with the block robustness of Steg, ensures that $\mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_2^*) = \sigma_t$ with probability at least $1 - \mathrm{negl}(\lambda)$. Conditioned on this event, the closeness of the first half $\Phi(\zeta_1^*, y_t) = 1$, together with the robustness of RDS, ensures that $\mathsf{RDS.Ver}(\mathsf{vk}, \zeta_1^*, \sigma_t) = 1$ with probability at least $1 - \mathrm{negl}(\lambda)$.

If both events occur, then $\mathsf{Ver}(\mathsf{vk}, \zeta) = 1$, since it accepts if *any* $2n$-substring of ζ verifies under this two-part check. Because both events occur with all but negligible probability, successful verification also occurs with all but negligible probability.

Unforgeability. Let Π denote the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$. Suppose, towards contradiction, that there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that

$$\Pr\left[\begin{array}{r} \mathsf{Ver}(\mathsf{vk}, \zeta) = 1 \\ \wedge \neg(\exists \zeta^* \preceq \zeta, (x, y) \in \Pi \text{ such that} \\ \mathsf{EBC}[\Phi, n](\zeta^*, y) = 1 \ \wedge \ |\zeta^*| = n) \end{array}\right] \geq 1/q(\lambda) ,$$

When the above event occurs, then for every substring $\zeta^* \preceq \zeta$ of size n, and for every $(x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$, ζ^* is $\mathsf{EBC}[\Phi, n]$-far from y, which ensures that for every block index $t \in [\lfloor |y|/n \rfloor]$, $\Phi(\zeta^*, y_t) = 0$.

Note that the set of all such blocks y_t subsumes the set of message for which $\mathcal{A}$ observed robust signatures, because the only messages that $\mathcal{A}$ sees signatures for during interaction with $\mathsf{Wat}_{\mathsf{sk}}$ are response blocks y_t. This means that for all $\zeta^* \preceq \zeta$ such that $|\zeta^*| = n$, and for all $(y, \cdot) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RDS.Sign}_{\mathsf{RDS.sk}}})$, $\Phi(\zeta^*, y) = 0$.

However, under the same event, $\mathsf{Ver}(\mathsf{vk}, \zeta) = 1$, meaning $\mathcal{A}$ found some $\zeta^* \preceq \zeta$ of size n and some $\sigma \in \{0,1\}^k$ such that $\mathsf{RDS.Ver}(\mathsf{RDS.pk}, \zeta^*, \sigma) = 1$ with probability at least $1/q(\lambda)$. These two facts combined contradict the unforgeability of RDS.

□

6.2 Generalizing verification to chains

The verification algorithm for our watermarking scheme (Figure 5) applies a sliding window to ζ, checking whether any substring of size $2n$ contains a valid message–signature pair. As a result, Theorem 6.1 guarantees unforgeability with respect to substrings of size n (the size of the signed message).

We can generalize this approach by using a sliding window of size rn for any $r \in \mathbb{N}$ with $r \geq 2$. Within each rn-bit window, we can verify that the *entire chain* of r, n-bit blocks verifies back-to-back. This yields a stronger unforgeability guarantee: successful verification implies the existence of an $(r-1)n$-bit substring in ζ that is close to a consecutive *chain* of blocks, $(y_t, \ldots, y_{t+r-2})$, present in a single observed response. We formally define a family of general verification algorithms, $\{\mathsf{Ver}_r\}_{r \in \mathbb{N}}$, in Figure 6.

Algorithm 12 Ver_r

```
input: (vk, ζ)
n ← n(λ)                                          // compute block size n
for i ∈ [|ζ| − rn + 1] do
    b ← ChainVer_r(vk, ζ[i : i + rn])             // verify the entire chain
    if b = 1 then
        return 1
    end if
end for
return 0
```

Chain verification takes r blocks of size n as input and verifies that for each consecutive pair of blocks $(\zeta_j^*, \zeta_{j+1}^*)$, the signature recovered from the second block successfully verifies with the message given by the first. If any block fails verification, the entire chain is rejected.

Algorithm 13 $\mathsf{ChainVer}_r$

input: $(\mathsf{vk}, \zeta^* = (\zeta_1^*, \ldots, \zeta_r^*) \in \{0,1\}^{rn})$
$\mathsf{flag} \leftarrow 1$
for $j \in [r-1]$ **do**
  $\sigma_j \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_{j+1}^*)$ `// attempt to decode signature`
  if $\mathsf{RDS.Ver}(\mathsf{RDS.pk}, \zeta_j^*, \sigma_j) = 0$ **then**
    $\mathsf{flag} \leftarrow 0$ `// if any pair fails verification, set flag to 0`
  end if
end for
return flag `// indicates whether or not` *`every`* `block verified`

Figure 6: Pseudocode for chain verification.

Theorem 6.2 states the guarantees obtained when instead using Ver_r as the verifier for our construction. Its guarantees differ from those obtained using 2-block verification in two respects. First, chain verification requires that RDS has *unique signatures* (Definition 5.3), though, as noted previously, this is not particularly restrictive. Second, unlike in the 2-block case, whether unforgeability holds in the public or secret key setting additionally depends on the robustness guarantees of the underlying block steganography scheme Steg.

Theorem 6.2 (Robust and unforgeable r-block watermarking)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, let $r = \mathrm{poly}(\lambda)$ satisfying $r \geq 2$ be any chain size, and let $k, n = \mathrm{poly}(\lambda)$ be any message and block size, respectively. Let Φ be any closeness predicate, and Q be any language model. Suppose that:*

- $\mathsf{Steg}[\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec}]$ *is a message size k, block size n block steganography scheme for Q with (secret or public key) Φ-robustness (Definition 4.11).*
- $\mathsf{RDS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ *is a message size n, signature size k Φ-robust digital signature scheme (Definition 5.1) that has unique signatures w.r.t. Φ (Definition 5.3).*

Then the construction $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver}_r)$ (Figures 5 and 6) is a watermarking scheme for Q with (secret or public key) $(\mathsf{EBC}[\Phi, n], rn)$-robustness, and (secret or public key) $(\mathsf{EBC}[\Phi, n], (r-1)n)$-unforgeability.

Proof. Undetectability and robustness are straightforward from the robustness and undetectability of Steg and the robustness of RDS. Therefore, we will focus on proving the unforgeability guarantee. We will prove secret key unforgeability; public key follows analogously.

Fix $\lambda \in \mathbb{N}$ and a PPT adversary $\mathcal{A}$. Let

$$(\mathsf{RDS.pk}, \mathsf{RDS.sk}) \leftarrow \mathsf{RDS.Gen}(1^\lambda), \quad (\mathsf{Steg.vk}, \mathsf{Steg.sk}) \leftarrow \mathsf{Steg.Gen}(1^\lambda) \,.$$

Denote $\mathsf{vk} = (\mathsf{RDS.pk}, \mathsf{Steg.vk})$ and $\mathsf{sk} = (\mathsf{RDS.sk}, \mathsf{Steg.sk})$. Let Π denote the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$ and let $\zeta \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}, (\mathsf{Ver}_r)_{\mathsf{vk}}}(1^\lambda)$

We begin by showing that the event that a string ζ is $\mathsf{EBC}[\Phi, n]$-far from every response y implies a more convenient event. For a predicate Φ and an index $i \in \mathbb{N}$, define

$$\mathsf{SuffixFormsCloseChain}_{\Phi,i}(\zeta,\Pi) := \begin{pmatrix} \exists \ell \in \mathbb{N} \text{ such that } |\zeta| = \ell n \text{ and} \\ \Big(i > \ell \text{ or } \exists (x,y) \in \Pi \text{ such that} \\ \mathsf{EBC}[\Phi,n]\big((\zeta_i,\ldots,\zeta_\ell),\, y\big) = 1\Big) \end{pmatrix}. \tag{2}$$

Given a sequence $\zeta = (\zeta_1, \ldots, \zeta_\ell)$ of ℓ blocks of size n, the event asserts that the suffix starting at the ith block, $(\zeta_i, \ldots, \zeta_\ell)$ is $\mathsf{EBC}[\Phi, n]$-close to some response y appearing in the transcript Π. When $i > \ell$, we interpret $\mathsf{SuffixFormsCloseChain}_{\Phi,i}(\zeta,\, \Pi) = 1$.

Now notice that if a string $\zeta = (\zeta_1, \ldots, \zeta_\ell)$ is $\mathsf{EBC}[\Phi, n]$-*far* from all responses y, then there must be a point at which the blocks of ζ transition from being close to a response chain to not. Formally, there exists an index $i \in [\ell]$ such that the suffix $(\zeta_{i+1}, \ldots, \zeta_\ell)$ is close to some response, but $(\zeta_i, \ldots, \zeta_\ell)$ is not:

$$\Big(\forall (x,y) \in \Pi, \mathsf{EBC}[\Phi,n](\zeta,y) = 0\Big) \\ \subseteq \bigvee_{i \in [\ell]} \Big(\mathsf{SuffixFormsCloseChain}_{\Phi,i+1}(\zeta,\Pi) \;\wedge\; \neg \mathsf{SuffixFormsCloseChain}_{\Phi,i}(\zeta,\Pi)\Big).$$

This also covers the case when *no* suffix of ζ is close to a response chain. In particular, if no suffix of ζ is $\mathsf{EBC}[\Phi, n]$-close to a response y, then for all $i \in [\ell]$ we have $\mathsf{SuffixFormsCloseChain}_{\Phi,i}(\zeta, \Pi) = 0$, so $\neg\mathsf{SuffixFormsCloseChain}_{\Phi,i}(\zeta, \Pi) = 1$. Hence, the right-hand side still evaluates to 1 due to the $i = \ell$ clause, where $\mathsf{SuffixFormsCloseChain}_{\Phi,\ell+1}(\zeta, \Pi) = 1$ is trivially true.

Now consider $\mathsf{ChainForgery}_i$ for $i \in \mathbb{N}$, the event that there exists a substring $\zeta^* \preceq \zeta$ such that the closeness chain of all-but-the-last block of ζ^* breaks exactly at index i, yet $\mathsf{ChainVer}_r$ still accepts it. Formally,

$$\mathsf{ChainForgery}_i(\zeta,\Pi,\mathsf{vk}) := \begin{pmatrix} \exists\, \zeta^* \preceq \zeta \text{ of length } rn \text{ such that} \\ \mathsf{ChainVer}_r(\mathsf{vk},\zeta^*) = 1, \\ \text{and } \mathsf{SuffixFormsCloseChain}_{\Phi,i+1}((\zeta_1^*,\ldots,\zeta_{r-1}^*),\Pi), \\ \text{and } \neg\mathsf{SuffixFormsCloseChain}_{\Phi,i}((\zeta_1^*,\ldots,\zeta_{r-1}^*),\Pi). \end{pmatrix}. \tag{3}$$

Lemma 6.3 shows that for any fixed $i \in [r-1]$, this event occurs with probability at most $\mathrm{negl}(\lambda)$. Applying a union bound over all $i \in [r-1]$ shows that the event $\bigvee_{i\in[r-1]} \mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})$ also occurs with negligible probability.

We conclude by showing that this suffices to prove unforgeability. Consider the event that an adversary successfully forges the watermark:

$$\mathsf{Ver}_r(\mathsf{vk},\zeta) = 1 \;\wedge\; \Big(\forall\, \zeta' \preceq \zeta, (x,y) \in \Pi,\; |\zeta'| \neq (r-1)n \;\vee\; \mathsf{EBC}[\Phi,n](\zeta',y) = 0\Big).$$

By definition of Ver_r, if $\mathsf{Ver}_r(\mathsf{vk}, \zeta) = 1$, then there exists a substring $\zeta^* \preceq \zeta$ of length rn such that $\mathsf{ChainVer}_r(\mathsf{vk}, \zeta^*) = 1$. Moreover, if every $(r-1)n$-sized substring $\zeta' \preceq \zeta$ is $\mathsf{EBC}[\Phi, n]$-far from all responses, then the specific substring $(\zeta_1^*, \ldots, \zeta_{r-1}^*) \preceq \zeta$, must be far from all responses. By the previous chain-breaking argument, this implies that there exists an index $i \in [r-1]$ where the chain breaks. Hence, the event of a successful forgery implies the event $\bigvee_{i\in[r-1]} \mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})$, which we showed happens with negligible probability. □

Lemma 6.3. *Let $\lambda \in \mathbb{N}$ be a security parameter, let $r = \mathrm{poly}(\lambda)$ satisfying $r \geq 2$ be any chain size, and let $i \in [r-1]$ be any block index. Let $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Ver}_r)$ be the watermarking scheme presented in Figures 5 and 6. Under the same assumptions as Theorem 6.2, for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rcl}(\mathsf{pk},\mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow & \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}},(\mathsf{Ver}_r)_{\mathsf{vk}}} \end{array} \;:\; \mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})\right] \leq \mathrm{negl}(\lambda)\ ,$$

where Π is the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$ and $\mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})$ is defined as in 3.

Proof. Fix $\lambda \in \mathbb{N}$. The case $i = r-1$ follows similarly to the proof of Theorem 6.1, therefore, we will prove the lemma for a fixed $i \in [r-2]$. Assume, toward contradiction, that there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that

$$\Pr[\mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})] \geq 1/q(\lambda)$$

Notice that the event $\mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})$ implies the following event, specifically for the substring $(\zeta_i^*, \zeta_{i+1}^*)$:

$$\begin{pmatrix}\exists\, \zeta^* \preceq \zeta \text{ of size } 2n \text{ such that } \mathsf{RDS.Ver}\Big(\mathsf{RDS.pk}, \zeta_1^*, \mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_2^*)\Big) = 1, \\ \text{and } \exists (x,y) \in \Pi, t \in \mathbb{N} \text{ such that } \Phi(\zeta_2^*, y_t) = 1 \text{ and } \big(\Phi(\zeta_1^*, y_{t-1}) = 0 \text{ or } t = 1\big)\end{pmatrix}.$$

Consequently, the event above occurs with probability at least $1/q(\lambda)$. Moreover, when this event occurs, the fact that $\Phi(\zeta_2^*, y_t) = 1$ and the block robustness of Steg tell us that $\mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_2^*)$ decodes to the message σ_{t-1} embedded in y_t with all but negligible probability. Therefore, with inverse-polynomial probability, $\mathcal{A}$ finds a block $\zeta_1^* \in \{0,1\}^n$ and a message $\sigma_{t-1} \in \{0,1\}^k$ such that

$$(\sigma_{t-1}, y_t) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Steg.Embed}_{\mathsf{Steg.sk}}}) \text{ and } \mathsf{RDS.Ver}(\mathsf{RDS.pk}, \zeta_1^*, \sigma_{t-1}) = 1\ .$$

By definition of $\mathsf{Wat}_{\mathsf{sk}}$, σ_{t-1} is either a uniformly random string when $t = 1$, or the robust signature of the previous block y_{t-1} when $t > 1$. We analyze these two cases separately.

1. $\sigma_{t-1} \leftarrow \{0,1\}^k$ is a uniformly random string, independent of ζ_1^*. We have that

   $$\mathsf{RDS.Ver}(\mathsf{RDS.pk},\ \zeta_1^*,\ \sigma_{t-1}) = 1$$

   with inverse-polynomial probability. But then, there exists an adversary $\mathcal{B}^{\mathsf{RDS.Sign}_{\mathsf{sk}}}(\mathsf{RDS.vk})$ against the strong unforgeability of RDS that simulates $\mathcal{A}$ and outputs ζ_1^*, along with a uniformly random string $\sigma' \leftarrow \{0,1\}^k$. This verifies with inverse-polynomial probability, even though σ' is likely not any of the signatures seen by $\mathcal{B}$.

2. $\sigma_{t-1} \leftarrow \mathsf{RDS.Sign}(\mathsf{RDS.sk}, y_{t-1})$ is a robust signature of the previous block. In this case,

   $$(y_{t-1}, \sigma_{t-1}) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RDS.Sign}_{\mathsf{RDS.pk}}}). \tag{4}$$

   However, we have that $\Phi(\zeta_1^*, y_{t-1}) = 0$. Either $(\zeta_1^*, \sigma_{t-1})$ contradicts the unforgeability of RDS, or there exists a different message $m \neq y_{t-1}$ such that

   $$(m, \sigma_{t-1}) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RDS.Sign}_{\mathsf{RDS.pk}}}) \text{ and } \Phi(\zeta_1^*, m) = 1\ .$$

   This, however, breaks the unique signature property of RDS, because the adversary finds two different messages $m \neq y_{t-1}$ with the same signature σ_{t-1}, where ζ_1^* is Φ-far from y_{t-1} but Φ-close to m. Thus, in the case, $\mathcal{A}$ either breaks unforgeability or unique signatures of RDS.

In both cases, we obtain a contradiction to the security of RDS. Therefore, $\mathsf{ChainForgery}_i(\zeta, \Pi, \mathsf{vk})$ occurs only with negligible probability. □

6.3 Recoverable watermarking

If the robust signature scheme RDS is recoverable, meaning it includes a recovery algorithm RDS.Rec that can reconstruct the original message from its signature and a perturbed version of that message, then we can recover the original prefix of the errored text using Rec. Notably, Rec does not rely on the underlying language model Q or generation prompt x. Consequently, it enables users to recover original content using only the verification key and modified text. We obtain the following guarantees:

- $(\mathsf{EBC}[\Phi, n], 2n, n)$-robust recoverability: if an (adversarially chosen) ζ contains a substring $\zeta^* \preceq \zeta$ of size rn for some $r \geq 2$, where the blocks $(\zeta_1^*, \ldots, \zeta_r^*)$ are blockwise Φ-close to r consecutive response blocks $(y_t, \ldots, y_{t+r-1})$, then $\mathsf{Rec}(\mathsf{vk}, \zeta)$ will recover a substring of that response, $\xi \preceq y$ of size $(r-1)n$. In particular, in our construction, $\xi = (y_t, \ldots, y_{t+r-2})$.
- $(\mathsf{EBC}[\Phi, n], n)$-unforgeable recoverability: if an (adversarially chosen) ζ leads to a recovery $\xi \in \mathsf{Rec}(\mathsf{vk}, \zeta)$ then ξ must satisfy:
  1. ξ has size rn for some $r \geq 1$,
  2. ξ is a substring of some close response y, i.e., $\xi \preceq y$, and there exists $\zeta^* \preceq \zeta$ of size rn that is $\mathsf{EBC}[\Phi, n]$-close to y.

  In particular, in our construction, ζ^* is $\mathsf{EBC}[\Phi, n]$-close to ξ and ξ forms chain of blocks in y.

Our recovery algorithm leverages the chaining idea from the previous section to not only reconstruct individual n-bit blocks using RDS.Rec, but to also recover multi-block prefixes when the adversary has approximately copied a *chain* of consecutive watermarked blocks. The idea is to apply chain recoverability—analogous to chain verification—with *every* window size rn, ensuring recovery from any long enough substring $\zeta^* \preceq \zeta$ with $|\zeta^*| \geq 2n$.

Algorithm 14 Rec

input: (vk, ζ)
$n \leftarrow n(\lambda)$ `// compute block size n`
$\mathcal{L} \leftarrow \emptyset$
for $r \in \{2, \ldots, \lfloor |\zeta|/n \rfloor\}$ `// for every chain size` r **do**
  for $i \in [|\zeta| - rn + 1]$ `// for every` rn`-bit substring in the input` **do**
    $\xi \leftarrow \mathsf{ChainRec}_r(\mathsf{vk}, \zeta[i : i + rn])$ `// attempt to recover the first` $r-1$ `blocks`
    if $\perp \not\preceq \xi$ and $|\xi| = (r-1)n$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{\xi\}$ `// only keep` ξ `if all` $r-1$ `recoveries were successful`
    end if
  end for
end for
return $\mathcal{L}$

The subroutine $\mathsf{ChainRec}_r$ takes in an r-block chain and attempts to recover the first $r-1$ blocks.

Algorithm 15 $\mathsf{ChainRec}_r$

input: $(\mathsf{vk}, \zeta^* = (\zeta_1^*, \ldots, \zeta_r^*))$
$n \leftarrow n(\lambda)$ `// compute block size n`
for $i \in [r-1]$ **do**
  $\sigma_j \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_{j+1}^*)$
  $\xi_j \leftarrow \mathsf{RDS.Rec}(\mathsf{RDS.pk}, \zeta_j^*, \sigma_j)$
end for
return $\xi = (\xi_1, \ldots, \xi_{r-1})$ `// may contain` $\perp$ `symbols`

Figure 7: Pseudocode for the recovery algorithm Rec, which calls upon $\mathsf{ChainRec}_r$ as a subroutine.

Corollary 6.4 (Recoverable watermarking)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter and let $k, n = \mathrm{poly}(\lambda)$ be any message and block size, respectively. Let Φ be any closeness predicate, and Q be any language model. Suppose that:*

- $\mathsf{Steg}[\mathsf{Gen}, \mathsf{Embed}, \mathsf{Dec}]$ *is a message size k, block size n block steganography scheme for Q with (secret or public key) Φ-robustness (Definition 4.11).*
- $\mathsf{RDS}[\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}]$ *is a message size n, signature size k Φ-recoverable digital signature scheme (Definition 5.2) that has unique signatures w.r.t. Φ (Definition 5.3).*

Then the construction $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Rec})$ *in Figures 5 and 7 is a* recoverable *watermarking scheme with (secret or public key)* $(\mathsf{EBC}[\Phi, n], 2n, n)$*-robustness and (secret or public key)* $(\mathsf{EBC}[\Phi, n], n)$*-unforgeability.*

Proof. Fix $\lambda \in \mathbb{N}$ and a PPT adversary $\mathcal{A}$. Let

$$(\mathsf{RDS.pk}, \mathsf{RDS.sk}) \leftarrow \mathsf{RDS.Gen}(1^\lambda), \quad (\mathsf{Steg.vk}, \mathsf{Steg.sk}) \leftarrow \mathsf{Steg.Gen}(1^\lambda) .$$

Denote $\mathsf{vk} = (\mathsf{RDS.pk}, \mathsf{Steg.vk})$ and $\mathsf{sk} = (\mathsf{RDS.sk}, \mathsf{Steg.sk})$. Let $\zeta \leftarrow \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\mathsf{vk})$. We prove the public key version for both properties; the secret key version follow analogously.

Robust recoverability. Suppose there exists a substring $\zeta^* \preceq \zeta$ with $|\zeta^*| \geq 2n$ and a pair $(x, y) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$ such that ζ^* and y are close under $\mathsf{EBC}[\Phi, n]$. Then there exist $r, t \in \mathbb{N}$, with $r \geq 2$, such that $|\zeta^*| = rn$ and

$$\Phi(\zeta_j^*, y_{t+j-1}) = 1, \ \forall j \in [r] .$$

Fix $j \in [r-1]$. As in the proof of robustness,

$$\sigma_j \leftarrow \mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_{j+1}^*)$$

is the signature of y_{t+j-1} with probability at least $1 - \mathrm{negl}(\lambda)$. Since $\Phi(\zeta_j^*, y_{t+j-1}) = 1$, the recovery algorithm of RDS satisfies $\mathsf{RDS.Rec}(\mathsf{RDS.pk}, \zeta_j^*, \sigma_j) = y_{t+j-1}$ with probability at least $1 - \mathrm{negl}(\lambda)$. Applying a union bound over all $j \in [r-1]$, we conclude that

$$\mathsf{ChainRec}_r(\mathsf{vk}, \zeta^*) = (y_t, \ldots, y_{t+r-2}),$$

which is a substring of y of size $|\zeta^*| - n$, with all but negligible probability. Since $r \leq \lfloor |\zeta|/n \rfloor$ (otherwise $rn > |\zeta|$ and ζ^* could not be a substring of ζ), $\mathsf{Rec}(\mathsf{vk}, \zeta)$ calls $\mathsf{ChainRec}_r(\mathsf{vk}, \zeta^*)$ during its execution, so the chain $(y_t, \ldots, y_{t+r-2})$, which is an $(|\zeta^*| - n)$-sized substring of y, will be added to the final output list with all but negligible probability.

Unforgeable recoverability. The proof follows similarly to the proof of unforgeable chain verification. Let Π be the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$. For a predicate Φ and a block index $i \in \mathbb{N}$, define

$$\mathsf{SuffixFormsExactChain}_{\Phi,i}(\xi, \Pi) := \begin{pmatrix} \exists (x, y) \in \Pi, t \in \mathbb{N} \text{ such that} \\ \xi_j = y_{t+j-i},\ \forall j \in \{i, \ldots, \ell\} \end{pmatrix}.$$

the event that an input string $\xi \in \{0,1\}^*$ forms an *exact* chain of blocks in some response y (in contrast with $\mathsf{SuffixFormsCloseChain}$ 2 from the chain verification proof, which only required the input was *close* to a chain of blocks in some response y). Moreover, define

$$\mathsf{SuffixRecoversChain}_{\Phi,i}(\zeta, \xi, \Pi) := \begin{pmatrix} \exists\, \ell \in \mathbb{N} \text{ such that } |\zeta| = |\xi| = \ell n \text{ and} \\ \Big(i > \ell \text{ or} \\ (\mathsf{EBC}[\Phi, n]((\zeta_i, \ldots, \zeta_\ell), (\xi_i, \ldots, \xi_\ell)) = 1 \\ \text{and } \mathsf{SuffixFormsExactChain}_{\Phi,i}(\xi, \Pi)) \Big) \end{pmatrix}.$$

Given two sequences $\zeta, \xi \in \{0,1\}^{\ell n}$ of ℓ blocks of size n, this event asserts that the suffix starting from the ith block $(\zeta_i, \ldots, \zeta_\ell)$ is blockwise Φ-close to the blocks $(\xi_i, \ldots, \xi_\ell)$, and that ξ forms of chain of blocks in some response y. When $i = 1$, this implies that $\xi \preceq y$ for some response y and that ζ is $\mathsf{EBC}[\Phi, n]$ close to y.

Now notice that if a string $\zeta = (\zeta_1, \ldots, \zeta_\ell)$ and recovery $\xi = (\xi_1, \ldots, \xi_\ell)$ do not form a recovery chain—that is, either ξ does not form a chain of blocks in some response y, or ζ is $\mathsf{EBC}[\Phi, n]$-far from ξ—then there must be some index $i \in [\ell]$ at which the chain transitions from being a valid recovery chain to not:

$$\Big(\mathsf{EBC}[\Phi, n](\zeta, \xi) = 0 \ \vee\ \neg \mathsf{SuffixFormsExactChain}_{\Phi,1}(\xi, \Pi)\Big) \\ \subseteq \bigvee_{i \in [\ell]} \Big(\mathsf{SuffixRecoversChain}_{\Phi,i+1}(\zeta, \xi, \Pi) \ \wedge\ \neg \mathsf{SuffixRecoversChain}_{\Phi,i}(\zeta, \xi, \Pi)\Big).$$

Now, for $i, r \in \mathbb{N}$, consider the event $\mathsf{ChainRecForgery}_{i,r}$: there exists a substring $\zeta^* \preceq \zeta$ such that $\mathsf{ChainRec}_r(\mathsf{vk}, \zeta^*)$ successfully recovers some string ξ, but for which the recovery chain between ζ^* and ξ breaks at index i. Formally,

$$\mathsf{ChainRecForgery}_{i,r}(\zeta, \Pi, \mathsf{vk}) := \begin{pmatrix} \exists\, \zeta^* \preceq \zeta \text{ of length } rn \text{ such that} \\ \mathsf{ChainRec}_r(\mathsf{vk}, \zeta^*) = \xi \ \text{ with } \ \perp \not\preceq \xi, \\ \text{and } \mathsf{SuffixRecoversChain}_{\Phi,i+1}((\zeta_1^*, \ldots, \zeta_{r-1}^*), \xi, \Pi), \\ \text{and } \neg \mathsf{SuffixRecoversChain}_{\Phi,i}((\zeta_1^*, \ldots, \zeta_{r-1}^*), \xi, \Pi) \end{pmatrix}. \tag{5}$$

Lemma 6.5 shows that for any fixed $r = \mathrm{poly}(\lambda)$ with $r \geq 2$ and $i \in [r-1]$, this event occurs with probability at most $\mathrm{negl}(\lambda)$. Applying a union bound over all $r \in \{2, \ldots, \lfloor |\zeta|/n \rfloor\}, i \in [r-1]$ shows that the event $\bigvee_{i,r} \mathsf{ChainRecForgery}_{i,r}(\zeta, \Pi, \mathsf{vk})$ also occurs with negligible probability.

We conclude by showing that this suffices to prove unforgeable recoverability. Consider the event that an adversary successfully forges:

$$\xi \in \mathsf{Rec}(\mathsf{vk}, \zeta) \ \wedge\ \Big(\forall\, \zeta' \preceq \zeta, (x, y) \in \Pi,\ |\zeta'| \neq |\xi| \ \vee\ \xi \not\preceq y \ \vee\ \mathsf{EBC}[\Phi, n](\zeta', y) = 0\Big).$$

By definition of Rec, if $\xi \in \mathsf{Rec}(\mathsf{vk}, \zeta)$, then there exists an $r \in \{2, \ldots, \lfloor |\zeta|/n \rfloor\}$ and substring $\zeta^* \preceq \zeta$ of size rn where $\mathsf{ChainVer}_r(\mathsf{vk}, \zeta^*) = \xi$, $\perp \not\preceq \xi$, and $|\xi| = (r-1)n$. Moreover, if for every $|\xi|$-sized substring $\zeta' \preceq \zeta$ and every response y, either $\xi \not\preceq y$ or ζ' is $\mathsf{EBC}[\Phi, n]$-far from y, then either ζ' is $\mathsf{EBC}[\Phi, n]$-far from ξ or ξ does not form a chain in any response.

By the previous chain breaking argument, this means that the suffix recovery chain must break at some point for every substring ζ' of size $|\xi|$, including the specific substring $(\zeta_1^*, \ldots, \zeta_{r-1}^*)$. Hence, the event of a successful forgery implies the event $\bigvee_{i,r} \mathsf{ChainRecForgery}_{i,r}(\zeta, \Pi, \mathsf{vk})$, which we showed happens with negligible probability. □

Lemma 6.5. *Let $\lambda \in \mathbb{N}$ be any security parameter, let $r = \mathrm{poly}(\lambda)$ satisfying $r \geq 2$ be any chain size, and let $i \in [r-1]$ be any block index. Let $(\mathsf{Gen}, \mathsf{Wat}, \mathsf{Rec})$ be the watermarking scheme presented in Figures 5 and 7. Under the same assumptions as Corollary 6.4, for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rcl} (\mathsf{pk}, \mathsf{sk}) & \leftarrow & \mathsf{Gen}(1^\lambda) \\ \zeta & \leftarrow & \mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}, \mathsf{Rec}_{\mathsf{vk}}} \end{array} : \mathsf{ChainRecForgery}_{i,r}(\zeta, \Pi, \mathsf{vk})\right] \leq \mathrm{negl}(\lambda) ,$$

where Π is the transcript $\mathsf{Trans}(\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}})$ and $\mathsf{ChainRecForgery}_{i,r}(\zeta, \Pi, \mathsf{vk})$ is defined as in 5.

Proof. Fix $\lambda \in \mathbb{N}$. The case $i = r-1$ is similar to the proof of Theorem 6.1, but by using the unforgeable recoverability of RDS instead of standard unforgeability. Hence, we prove the lemma for a fixed $i \in [r-2]$. Assume, toward contradiction, that there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that

$$\Pr[\mathsf{ChainRecForgery}_{i,r}(\zeta, \Pi, \mathsf{vk})] \geq 1/q(\lambda) .$$

Notice that this event implies the following event, specifically for the substring $(\zeta_i^*, \zeta_{i+1}^*)$:

$$\left(\begin{array}{l} \exists\, \zeta^* \preceq \zeta \text{ of size } 2n \text{ such that} \\ \mathsf{RDS.Rec}\Big(\mathsf{RDS.pk}, \zeta_1^*, \mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_2^*)\Big) = \xi \in \{0,1\}^n, \\ \text{and } \exists (x, y) \in \Pi, t \in \mathbb{N} \text{ such that} \\ \Phi(\zeta_2^*, y_t) = 1 \text{ and } \Big(\Phi(\zeta_1^*, \xi) = 0 \text{ or } \xi \neq y_{t-1} \text{ or } t = 1\Big) \end{array}\right) . \tag{6}$$

Consequently, the event above occurs with probability at least $1/q(\lambda)$. Moreover, when this event occurs, the fact that $\Phi(\zeta_2^*, y_t) = 1$ and the block-robustness of Steg tell us that $\mathsf{Steg.Dec}(\mathsf{Steg.vk}, \zeta_2^*)$ decodes to the message encoded in y_t with all but negligible probability. Therefore, with inverse-polynomial probability, $\mathcal{A}$ finds a block $\zeta_1^* \in \{0,1\}^n$ and a message $\sigma_{t-1} \in \{0,1\}^k$ such that

$$(\sigma_{t-1}, y_t) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{Steg.Embed}_{\mathsf{Steg.sk}}}) \text{ and } \mathsf{RDS.Rec}(\mathsf{RDS.pk}, \zeta_1^*, \sigma_{t-1}) = \xi \in \{0,1\}^n .$$

By definition of $\mathsf{Wat}_{\mathsf{sk}}$, the value σ_{t-1} is either a uniformly random string when $t = 1$, or the robust signature of the previous block y_{t-1} when $t > 1$. We analyze these two cases separately.

1. $\sigma_{t-1} \leftarrow \{0,1\}^k$ is a uniformly random string. We have that

   $$\mathsf{RDS.Rec}(\mathsf{RDS.pk}, \zeta_1^*, \sigma_{t-1}) \neq \perp$$

   with inverse-polynomial probability. As in the proof of chain verification, there exists an adversary against the unforgeable recovery of RDS.

2. $\sigma_{t-1} \leftarrow \mathsf{RDS.Sign}(\mathsf{RDS.pk}, y_{t-1})$ is a robust signature of the previous block. In this case,

$$(y_{t-1}, \sigma_{t-1}) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RDS.Sign}_{\mathsf{RDS.pk}}}) \ .$$

Since $\mathsf{RDS.Rec}(\mathsf{RDS.pk}, \zeta_1^*, \sigma_{t-1}) = \xi \in \{0,1\}^n$, it must be that

$$(\xi, \sigma_{t-1}) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RDS.Sign}_{\mathsf{RDS.pk}}}) \text{ and } \Phi(\zeta_1^*, \xi) = 1 \ . \tag{7}$$

with all but negligible probability; otherwise, $\mathcal{A}$ would break the unforgeable recoverability of RDS. Therefore, we may further condition on this event.

Together event 6, which guarantees that either $\Phi(\zeta_1^*, y_{t-1}) = 0$ or $\xi \neq y_{t-1}$, and event 7, which guarantees that $\Phi(\zeta_1^*, \xi) = 1$, imply that $\xi \neq y_{t-1}$; otherwise, ζ_1^* would be simultaneously Φ-close to and Φ-far from ξ, a contradiction.

When $\xi \neq y_{t-1}$, we have two subcases:

(a) If $\Phi(\zeta_1^*, y_{t-1}) = 1$, then with inverse-polynomial probability, $\mathsf{RDS.Rec}(\mathsf{vk}, \zeta_1^*, \sigma_{t-1}) \neq y_{t-1}$ despite the fact that $(y_{t-1}, \sigma_{t-1}) \in \mathsf{Trans}(\mathcal{A}^{\mathsf{RDS.Sign}_{\mathsf{RDS.sk}}})$ and $\Phi(\zeta_1^*, y_{t-1}) = 1$. This contradicts the robust recoverability of RDS.

(b) If $\Phi(\zeta_1^*, y_{t-1}) = 0$, then $\mathcal{A}$ produces $(\zeta_1^*, \xi, y_{t-1})$ that violates the unique signature property of RDS.

In both cases, we arrive at a contradiction, proving that the scheme is in fact unforgeably recoverable. □

6.4 Instantiating Hamming watermarks

We instantiate our watermarking scheme for high-entropy language models. In particular, we combine our RDS construction (Corollary 5.11) with an existing secret key block steganography scheme (Theorem 4.14) to obtain a watermark that is secret key recoverable, robust, and unforgeable in the presence of a constant fraction of Hamming errors.

Corollary 6.6 (Concrete secret-key watermarking)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and suppose there exists*

- *A CRHF with output size $m = m(\lambda)$.*
- *A strongly unforgeable digital signature with signature size $k = k(\lambda)$.*
- *An ideal PRC with with error rate $\delta_{\mathsf{PRC}} \in (0, \frac{1}{4})$ and information rate $\rho \in (0,1)$*[13]*.*

Let $n = \mathrm{poly}(\lambda)$ such that $n \geq \lambda$ and $(m+k)/n = o(1)$ be the block size. Let $\epsilon < \delta_{\mathsf{PRC}}$ be any constant and let Q be any language model satisfying the following entropy condition,

$$\forall \pi \in \{0,1\}^*, y \leftarrow \overline{Q}_n(\pi) \text{ satisfies } H_\infty(y) \geq 4\Big(\sqrt{\frac{1}{2} - \epsilon}\Big)n \ .$$

Let $\delta > 0$ be any constant such that

$$\delta < \min\left\{\delta_{\mathsf{RDS}}(\rho), \delta_{\mathsf{PRC}} - \epsilon\right\} \ ,$$

[13] Recall that for any error-rate $\delta_{\mathsf{PRC}} \in (0, 1/4)$, there exists an ideal PRC with constant rate $\rho = \Omega(1)$, assuming subexponential hardness of LPN [AACD+24, Theorem 3].

where $\delta_{\mathsf{RDS}}(\rho) := \sup\{\delta \in (0, 1/2) \mid \rho > H_2^{\mathsf{BZ}}(\delta)\}$. *Then, our construction in Corollary 6.4 yields a secret key*

$$(\mathsf{EBC}[\mathsf{Ham}_\delta, n], 2n)\text{-}\textit{robust and } (\mathsf{EBC}[\mathsf{Ham}_\delta, n], n)\text{-}\textit{unforgeable}$$

recoverable *watermarking scheme for* Q.

Proof. From the ideal PRC assumption and Theorem 4.14, for any error rate $\delta \leq \delta_{\mathsf{PRC}} - \epsilon$, there exists a block steganography scheme for Q with block size n, message size ρn. From the CRHF and strongly unforgeable digital signature assumptions and Corollary 5.11, for any $\delta \leq \delta_{\mathsf{RDS}}(\rho)$, there exists a Ham_δ-recoverable digital signature scheme with message size n and signature size $\rho n + o(n)$. We apply Theorem 6.1 and Corollary 6.4 to get the result. Note that the block steganography message size and the digital signature size differ by $o(n)$. Our proof ignores this difference, but it could be made exact by assuming an ideal PRC with a slightly better information rate $\rho' > \rho$. □

Moreover, we leverage an existing public-key block steganography scheme (Theorem 4.16) to obtain a public-key construction that is robust to a subconstant fraction of Hamming errors, subject to the restriction that the first ℓ bits of each block are error-free, as captured by the predicate Φ_δ defined in Theorem 4.16.

Corollary 6.7 (Concrete public-key watermarking)**.** *Let* $\lambda \in \mathbb{N}$ *be the security parameter, and suppose there exists*

- *A CRHF with output size* $m = m(\lambda)$.
- *A strongly unforgeable digital signature with signature size* $k = k(\lambda)$.

Let Q *be any language model such that for some* $\ell = \mathrm{poly}(\lambda)$,

$$\forall \pi \in \{0,1\}^*, y \leftarrow \overline{Q}_\ell(\pi) \text{ satisfies } H_\infty(y) \geq \omega(\log \lambda)\ .$$

Let $n = \mathrm{poly}(\lambda)$ *be any block size satisfying* $\ell(m + k)/n = o(1)$. *There exists a subconstant* $\delta_{\max} = \Theta(\frac{1}{\ell \log n})$ *such that for any* $\delta = o(1)$ *satisfying* $\delta(\lambda) \leq \delta_{\max}(\lambda)\ \forall \lambda \in \mathbb{N}$, *there exists a public key*

$$(\mathsf{EBC}[\Phi_\delta, n], 2n)\text{-}\textit{robust and } (\mathsf{EBC}[\mathsf{Ham}_\delta, n], n)\text{-}\textit{unforgeable}$$

recoverable *watermarking scheme for* Q *in the random oracle model, where* Φ_δ *is defined the same as in Theorem 4.16.*

Proof. Fix $\lambda \in \mathbb{N}$ sufficiently large. By Theorem 4.16, in the random oracle model there exist constants $\rho \in (0, 1)$ and $\delta^* \in (0, 1/2)$ such that for any $\delta \leq \delta^*$, there exists a public key Φ_δ-robust block steganography scheme for Q with block size n and message size $\rho(n/\ell - 1)$.

Moreover, the CRHF and strongly unforgeable digital signature assumptions, Corollary 5.13 guarantees that for any $\delta = o(1)$, there exists a Ham_δ-robust digital signature scheme with signature size

$$O(\delta n \log n) + m + k \leq C\delta n \log n + o(n/\ell)$$

for some constant $C > 0$.

To apply our main theorem, we require that the signature fits as a message for the block steganography scheme:

$$C\delta n \log n + o(n/\ell) \leq \rho(n/\ell - 1)\ .$$

This condition holds whenever

$$\delta(\lambda) \leq \delta_{\max}(\lambda) \text{ where } \delta_{\max} = \frac{\rho(n/\ell) - \rho - o(n/\ell)}{Cn \log n} = \Theta(\frac{1}{\ell \log n})$$

Therefore, there exists a $\mathsf{Ham}_{\delta(\lambda)}$-recoverable signature scheme with message size n and signature size at most $\rho(n/\ell - 1)$. Note we can adjust $\delta_{\max}$ so that additionally, $\delta(\lambda) \leq \delta^*$ for all λ, since δ^* is constant. Then both a Φ_δ-robust block steganography scheme and a Ham_δ-recoverable signature scheme exist.

Furthermore, for any strings $y, y' \in \{0,1\}^*$, we have

$$\Phi_\delta(y, y') = 1 \implies \mathsf{Ham}_\delta(y, y') = 1,$$

because if at most $\delta(\frac{n}{\ell} - 1)$ blocks of size ℓ differ between inputs y and y', then at most $\delta(n - \ell) \leq \delta n$ bits differ. It follows that the robust signature is also Φ_δ-robustly recoverable. Applying Theorem 6.1 with these constructions then yields the desired result. □

Acknowledgements

This work was supported in part by the Simons Collaboration on the Theory of Algorithmic Fairness. Part of this work was conducted while some of the authors were visiting the Simons Institute for the Theory of Computing.

References

[AAAA+23] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. "GPT-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023) (page 3).

[AACD+24] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. "Ideal Pseudorandom Codes". In: *arXiv preprint arXiv:2411.05947* (2024) (pages 4, 9, 11, 13, 15, 18, 25, 26, 30, 39, 50).

[Aar22] Scott Aaronson. *My AI Safety Lecture for UT Effective Altruism*. https://scottaaronson.blog/?p=6823. Accessed Oct 2024. 2022 (pages 3, 12).

[ABNN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M Roth. "Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs". In: *IEEE Transactions on information theory* 38.2 (1992) (page 32).

[ARCH+01] Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. "Natural language watermarking: Design, analysis, and a proof-of-concept implementation". In: *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*. Springer. 2001 (page 12).

[ARHK+02] Mikhail J Atallah, Victor Raskin, Christian F Hempelmann, Mercan Karahan, Radu Sion, Umut Topkara, and Katrina E Triezenberg. "Natural language watermarking and tamperproofing". In: *International workshop on information hiding*. Springer. 2002 (page 12).

[BBBC+23] Clark Barrett, Brad Boyd, Elie Bursztein, Nicholas Carlini, Brad Chen, Jihye Choi, Amrita Roy Chowdhury, Mihai Christodorescu, Anupam Datta, Soheil Feizi, et al. "Identifying and mitigating the security risks of generative AI". In: *Foundations and Trends® in Privacy and Security* 6.1 (2023) (page 3).

[BLV19] Elette Boyle, Rio LaVigne, and Vinod Vaikuntanathan. "Adversarially Robust Property-Preserving Hash Functions". In: *10th Innovations in Theoretical Computer Science* (2019) (pages 6, 11, 13, 26, 34).

[BRVV25] Andrej Bogdanov, Alon Rosen, Neekon Vafa, and Vinod Vaikuntanathan. "Adaptive Robustness of Hypergrid Johnson-Lindenstrauss". In: *arXiv preprint arXiv:2504.09331* (2025) (pages 13, 19).

[BS07] Mihir Bellare and Sarah Shoup. "Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles". In: *International Workshop on Public Key Cryptography*. Springer. 2007 (pages 9, 12, 14, 24).

[BZ82] EL Blokh and Victor V Zyablov. "Linear concatenated codes". In: *Moscow, USSR: Nauka* (1982) (page 27).

[Cac98] Christian Cachin. "An information-theoretic model for steganography". In: *International Workshop on Information Hiding*. Springer. 1998 (page 12).

[CG24] Miranda Christ and Sam Gunn. "Pseudorandom error-correcting codes". In: *Annual International Cryptology Conference*. Springer. 2024 (pages 3–5, 9, 11–13, 15, 25, 28, 30, 31, 39).

[CGZ24] Miranda Christ, Sam Gunn, and Or Zamir. “Undetectable watermarks for language models”. In: *The Thirty Seventh Annual Conference on Learning Theory*. PMLR. 2024 (pages 3, 8, 12).

[CHS24] Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. “Watermarking Language Models for Many Adaptive Users”. In: *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society. 2024 (page 13).

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”. In: *SIAM Journal on Computing* 38.1 (2008) (pages 13, 14, 18, 25, 35, 36, 58, 59).

[FGJM+25] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. “Publicly-Detectable Watermarking for Language Models”. In: *IACR Communications in Cryptology* 1.4 (2025) (pages 3–5, 9, 11–13, 15, 28, 32, 39).

[FLS22] Nils Fleischhacker, Kasper Green Larsen, and Mark Simkin. “Property-preserving hash functions for hamming distance from standard assumptions”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2022 (pages 11, 13, 27, 34).

[FOVJ+25] Jaiden Fairoze, Guillermo Ortiz-Jimenez, Mel Vecerik, Somesh Jha, and Sven Gowal. “On the difficulty of constructing a robust and publicly-detectable watermark”. In: *arXiv preprint arXiv:2502.04901* (2025) (page 19).

[FS21] Nils Fleischhacker and Mark Simkin. “Robust property-preserving hash functions for hamming distance and more”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2021 (page 27).

[GBSS+25] Sven Gowal, Rudy Bunel, Florian Stimberg, David Stutz, Guillermo Ortiz-Jimenez, Christina Kouridi, Mel Vecerik, Jamie Hayes, Sylvestre-Alvise Rebuffi, Paul Bernard, et al. “SynthID-Image: Image watermarking at internet scale”. In: *arXiv preprint arXiv:2510.09263* (2025) (page 18).

[GG24] Surendra Ghentiyala and Venkatesan Guruswami. “New constructions of pseudorandom codes”. In: *arXiv preprint arXiv:2409.07580* (2024) (page 13).

[GM24] Noah Golowich and Ankur Moitra. “Edit Distance Robust Watermarks for Language Models”. In: *Advances in Neural Processing Systems (NeurIPS)* (2024) (pages 3, 4, 13, 28).

[GR08] Venkatesan Guruswami and Atri Rudra. “Better binary list decodable codes via multilevel concatenation”. In: *IEEE Transactions on Information Theory* 55.1 (2008) (page 27).

[GZS24] Sam Gunn, Xuandong Zhao, and Dawn Song. “An undetectable watermark for generative image models”. In: *International Conference on Learning Representations (ICLR)* (2024) (pages 3, 12, 13, 15, 21).

[HLTW22] Justin Holmgren, Minghao Liu, LaKyah Tyner, and Daniel Wichs. “Nearly optimal property preserving hashing”. In: *Annual International Cryptology Conference (CRYPTO)*. Springer. 2022 (pages 11–13, 25, 27, 34, 36, 39, 57, 58).

[HLV02] Nicholas J Hopper, John Langford, and Luis Von Ahn. “Provably secure steganography”. In: *Annual International Cryptology Conference*. Springer. 2002 (page 13).

[JKPT12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. "Commitments and efficient zero-knowledge proofs from learning parity with noise". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2012 (pages 11, 26).

[KGWK+23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. "A watermark for large language models". In: *International Conference on Machine Learning*. PMLR. 2023 (pages 3, 12).

[KGWS+23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. "On the reliability of watermarks for large language models". In: *arXiv preprint arXiv:2306.04634* (2023) (page 12).

[KTHL23] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. "Robust distortion-free watermarks for language models". In: *arXiv preprint arXiv:2307.15593* (2023) (page 12).

[LPLL+24] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. "A survey of text watermarking in the era of large language models". In: *ACM Computing Surveys* 57.2 (2024) (page 12).

[MPSW10] Silvio Micali, Chris Peikert, Madhu Sudan, and David A Wilson. "Optimal error correction for computationally bounded noise". In: *IEEE Transactions on Information Theory* 56.11 (2010) (page 13).

[NN90] Joseph Naor and Moni Naor. "Small-bias probability spaces: Efficient constructions and applications". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990 (page 32).

[RBLE+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022 (page 3).

[Sim84] Gustavus J Simmons. "The prisoners' problem and the subliminal channel". In: *Advances in cryptology: proceedings of crypto 83*. Springer. 1984 (page 12).

[SSZP+24] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. "AI models collapse when trained on recursively generated data". In: *Nature* 631.8022 (2024) (page 4).

[Ta-17] Amnon Ta-Shma. "Explicit, almost optimal, epsilon-balanced codes". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 2017 (page 32).

[TTD05] Mercan Topkara, Cuneyt M Taskiran, and Edward J Delp III. "Natural language watermarking". In: *Security, Steganography, and Watermarking of Multimedia Contents VII*. Vol. 5681. SPIE. 2005 (page 12).

[YLBS+24] Hiromu Yakura, Ezequiel Lopez-Lopez, Levin Brinkmann, Ignacio Serna, Prateek Gupta, Ivan Soraperra, and Iyad Rahwan. "Empirical evidence of Large Language Model's influence on human spoken communication". In: *arXiv preprint arXiv:2409.01754* (2024) (page 6).

[ZALW23] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. “Provable robust watermarking for ai-generated text”. In: *arXiv preprint arXiv:2306.17439* (2023) (page 12).

[Zam24] Or Zamir. “Excuse me, sir? Your language model is leaking (information)”. In: *arXiv preprint arXiv:2401.10360* (2024) (page 13).

[ZGCF+24] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, et al. “SoK: Watermarking for AI-Generated Content”. In: *arXiv preprint arXiv:2411.18479* (2024) (page 12).

[Zha25] Dongfang Zhao. “Privacy-Preserving Hamming Distance Computation with Property-Preserving Hashing”. In: *arXiv preprint arXiv:2503.17844* (2025) (page 13).

A Strong Difference Recovery and Homomorphic PPH

We notice that the PPH construction of Holmgren et al. [HLTW22], when instantiated with a *homomorphic* collision-resistant hash function (CRHF), satisfies two notable properties: it supports strong difference recovery and it is homomorphic. We formalize both properties and state the guarantees.

Note, however, that our recoverable watermarking scheme does not rely on strong recovering or homomorphic PPHs, as the perturbed input x' is available in the clear. Instead, the notion of *sharp sketches*, introduced in Section B, more accurately captures the relevant functionality. We include strong difference recovery and homomorphic PPHs here primarily for completeness.

Definition A.1 (Strong difference recovering PPH)**.** *A Φ-PPH* $(\mathsf{Sample}, \mathsf{Hash}, \mathsf{Eval})$ *is strong difference-recovering if there exists a PPT algorithm* Rec *such that for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} h & \leftarrow \mathsf{Sample}(1^\lambda) \\ (x, x') & \leftarrow \mathcal{A}(1^\lambda, h) \\ \Delta & \leftarrow \mathsf{Rec}(h, h(x), h(x')) \end{array} : \begin{array}{r} (\Delta \neq x \oplus x' \wedge \Phi(x, x') = 1) \\ \vee\ (\Delta \neq \perp \wedge \Phi(x, x') = 0) \end{array}\right] \leq \mathrm{negl}(\lambda)$$

Definition A.2 (Homomorphic PPH)**.** *A PPH scheme* $(\mathsf{Sample}, \mathsf{Hash}, \mathsf{Eval})$ *with input size n and hash size k is said to be* homomorphic *if there exists a fixed abelian group $(\mathcal{G}_n, +)$ and an injective map $\psi : \mathcal{G}_n \to \{0,1\}^k$ such that for every hash function representation $h \in \mathcal{H}$, there exists an efficiently computable*[14] *homomorphism $\varphi : \mathbb{Z}^n \to \mathcal{G}_n$ satisfying*

$$\mathsf{Hash}(h, x) = \psi(\varphi(x)) \quad \textit{for all } x \in \{0,1\}^n \ .$$

In other words, every hash function $h : \{0,1\}^n \to \{0,1\}^k$ in the family $\mathcal{H}$ can be expressed as the composition of a homomorphism $\varphi : \mathbb{Z}^n \to \mathcal{G}_n$ and a fixed injective embedding $\psi : \mathcal{G}_n \to \{0,1\}^k$.

Notation for homomorphic PPH. If $\mathsf{PPH}[\mathsf{Sample}, \mathsf{Hash}, \mathsf{Eval}]$ is homomorphic, we abuse notation and write $h(x) - h(x') = h(x - x')$ for any $h \in \mathcal{H}$ and $x, x' \in \{0,1\}^n$, where subtraction is interpreted over $\mathbb{Z}^n$ (so that $x - x' \in \{-1, 0, 1\}^n$) and $h(x)$ refers to the group element $\varphi(x) \in \mathcal{G}_n$ (i.e., before applying the embedding ψ into $\{0,1\}^k$).

Definition A.3 (Sparsity testing)**.** *A homomorphic PPH scheme* $(\mathsf{Sample}, \mathsf{Hash}, \mathsf{Eval})$ *is said to support* sparsity testing *if there exists an algorithm* $\mathsf{SparseEval}$ *such that for all $h \in \mathcal{H}$ and $x, x' \in \{0,1\}^n$,*

$$\begin{aligned} \mathsf{Eval}(h, h(x), h(x')) &= \mathsf{SparseEval}(h, h(x) - h(x')) \\ &= \mathsf{SparseEval}(h, h(\Delta)) \ , \end{aligned}$$

where $\Delta = x - x' \in \{-1, 0, 1\}^n$, and the final equality follows from the homomorphic property.

In other words, if a homomorphic PPH supports sparsity testing, then its evaluation algorithm Eval depends only on the hash function h and the difference $h(x) - h(x') = h(x - x')$, rather than on the individual hash values. Specifically, Eval returns 1 if the difference Δ is sparse, and 0 otherwise.

Holmgren et al.'s homomorphic PPH goes further by supporting *sparse recovery*: if Δ is sparse, then given only $(h, h(\Delta))$, their recovery algorithm Rec outputs $\Delta \in \{-1, 0, 1\}^n$ exactly. Note that this recovers only the difference vector, not the preimages x or x'.

[14] That is, computable using $\mathrm{poly}(n)$ many elementary algebraic operations.

Theorem A.4 ([HLTW22, Corollary 6.7])**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and let $n = \mathrm{poly}(\lambda)$ denote the input size. Suppose there exists a homomorphic CRHF with output size $\ell = \ell(\lambda)$ that is collision-resistant over $\{-1, 0, 1\}^n$ [HLTW22, Definition 6.1]. Then, for any constants $\delta \in (0, 1/2)$ and $\rho > H_3^{\mathsf{BZ}}(\delta) \cdot \log_2 3$, where $H_3^{\mathsf{BZ}}(\cdot)$ refers to the Blokh-Zyablov bound [HLTW22, Fact 2.12], there exists a* strongly recoverable *homomorphic PPH over the input domain $\{0,1\}^n$ for Ham_δ with hash size $\rho \cdot n + \ell$.*

B Sharp Sketches and Lower Bounds

A *sharp sketch* is a cryptographic primitive that enables efficient recovery given auxiliary input x' in the clear. The goal is to reconstruct an input x from its sketch $g(x)$ and a candidate x' that is sufficiently close to it. Its defining feature is *sharpness*: $\mathsf{Rec}(g, g(x), x')$ must output $\perp$ if x' is far from x.

This notion is inspired by the secure sketches of Dodis et al. [DORS08], but differs in both purpose and security guarantees. Secure sketches require *unconditional* hardness of inversion: it must be hard to recover x from its sketch $g(x)$, even for an unbounded adversary. This necessitates a non-trivial compression rate $\rho = k/n$, so secure sketches are compressive by definition. Their recoverability guarantee is also *unconditional*, requiring recovery to succeed for *every* x' that is close to x.

In contrast, sharp sketches enforce a different notion of security, one not centered on inversion. They require that no *computationally-bounded* adversary can produce input pairs (x, x') that violate the sharp recoverability condition (see Definition B.1). Thus, compressiveness is not required by definition. Even the identity sketch $g(x) = x$ qualifies as a (trivial) sharp sketch, though it would not qualify as a secure sketch. On the other hand, any secure sketch can be converted into a sharp sketch by returning $\perp$ whenever $\hat{x} = \mathsf{Rec}(g, g(x), x')$ is either inconsistent $g(x) \neq g(\hat{x})$ or far from x'.

Sharp sketch constructions for Hamming predicates were given by Dodis et al. [DORS08, Section 8.3]. We include it here for completeness and to enable precise reference in our watermarking scheme, where it plays a key role. Our main technical contribution is a generalization of the lower bound by Holmgren et al. [HLTW22], extending it from Hamming predicates to any predicate Φ that satisfies an efficient sampleability condition (Theorem B.5). When specialized to the Hamming case, our bound implies that the construction of Dodis et al. achieves nearly optimal sketch size, since the gap between the Hamming bound $H(\delta)$ and the Blokh-Zyablov bound $H_2^{\mathsf{BZ}}(\delta)$ is small (see also Remark 4.8). This shows that the recoverable digital signature in Corollary 5.11 is nearly optimal compared to constructions based on the generic PPH/sharp sketch transformation of Section 5.2.

Definition B.1 (Sharp sketch)**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, let Φ be any predicate, and let $n = n(\lambda)$ and $k = k(\lambda)$ be such that $k \leq n$. A sharp sketching scheme for Φ with input size n and sketch size k is a triple of PPT algorithms $(\mathsf{Sample}, \mathsf{Sketch}, \mathsf{Rec})$ defined as follows:*

- $\mathsf{Sample}(1^\lambda)$*: samples a sketching function g from a family of functions $\mathcal{G}_\lambda$.*
- $\mathsf{Sketch}(g, x)$*: takes in $x \in \{0,1\}^n$ and outputs its sketch $g(x) \in \{0,1\}^k$.*
- $\mathsf{Rec}(g, z, x')$*: takes in $z \in \{0,1\}^k$ and $x' \in \{0,1\}^n$, and returns an output in $\{0,1\}^n \cup \{\perp\}$.*

The scheme satisfies the following security guarantee:

Sharp recoverability: *For any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{rl} g & \leftarrow \mathsf{Sample}(1^\lambda); \\ (x, x') & \leftarrow \mathcal{A}(1^\lambda, g) \end{array} : \begin{array}{r} (\mathsf{Rec}(g, g(x), x') \neq x \ \wedge \ \Phi(x, x') = 1) \\ \vee \ (\mathsf{Rec}(g, g(x), x') \neq \perp \ \wedge \ \Phi(x, x') = 0) \end{array}\right] \leq \mathrm{negl}(\lambda) \ .$$

We restate Theorem 5.6, which gave a difference recovery hash family for Hamming from CRHFs, in the language of sharp sketches.

Theorem B.2 (Sharp sketch for Hamming, adapted from [DORS08, Section 8.3])**.** *Let $\lambda \in \mathbb{N}$ be the security parameter, and let $n = n(\lambda)$ denote the input size. Suppose there exists a CRHF with output size $\ell = \ell(\lambda)$. For any constants $\delta \in (0, 1/2)$ and $\rho > H_2^{\mathsf{BZ}}(\delta)$, where $H_2^{\mathsf{BZ}}(\cdot)$ refers to the Blokh-Zyablov bound, there exists a sharp sketching scheme $(\mathsf{Sample}, \mathsf{Sketch}, \mathsf{Rec})$ over $\{0,1\}^n$ for Ham_δ with sketch size $\rho \cdot n + \ell$.*

Remark B.3 (PPH and sharp sketch)**.** *By Proposition 5.5, every Hamming PPH yields a sharp sketch for the same predicate. However, the converse does not hold. For instance, the sharp sketching scheme in Theorem 5.6 does not give rise to a Hamming PPH, as it crucially relies on receiving x' in the clear, rather than just its sketch. Note, however, that our recoverable digital signature algorithm (Section 5.3)* does *receive one input in the clear, and therefore sharp sketches suffice.*

Lemma B.4 ([DORS08], Lemma 2.2)**.** *For random variables X, Y, Z, where Y is supported over a set of size k, we have*

$$H_\infty(X|Y,Z) \geq H_\infty(X|Z) - \log k$$

Theorem B.5 (Sharp sketch lower bound)**.** *Let Φ any predicate and let $A \subseteq \{(x,x') \in \{0,1\}^n \times \{0,1\}^n : \Phi(x,x') = 1\}$ be such that the uniform distribution $\mathcal{U}$ over pairs (x,x') conditioned on A is efficiently sampleable. Then the sketch size of any sharp sketch for Φ is at least*

$$n + \log \mathcal{U}(A) \ .$$

Proof. Let X be uniformly distributed over $\{0,1\}^n$, and let $g : \{0,1\}^n \to \{0,1\}^k$ be a sharp sketch drawn independently of X. Then the min-entropy of X conditioned on h and $g(X)$, which we denote by $H_\infty(X \mid h, g(X))$, satisfies:

$$H_\infty(X \mid g, g(X)) \geq H_\infty(X \mid g) - k \geq n - k \ .$$

In particular, no (even unbounded) adversary can "invert" $(g, g(X))$ to recover the original input X with probability greater than $2^{-(n-k)}$ when X is drawn uniformly from $\{0,1\}^n$.

Now consider the following inversion strategy based on $\mathsf{Rec}(g, g(x), \cdot)$. Define a randomized algorithm Inv that, on input $(g, g(x))$, samples a fresh $x' \leftarrow \{0,1\}^n$ and outputs $\mathsf{Rec}(g, g(x), x')$. Let μ be the distribution of (x,x') sampled uniformly from $\{0,1\}^{2n}$ conditioned on the event A. By assumption, $(x,x') \in A$ implies $\Phi(x,x') = 1$. Since μ is efficiently sampleable and $\Phi(x,x') = 1$, guarantees of the sharp sketch g yield

$$\Pr\left[\begin{array}{rl} g & \leftarrow \mathsf{Sample}(1^\lambda) \\ (x,x') & \leftarrow \mu \end{array} : \mathsf{Rec}(g, g(x), x') = x \right] \geq 1 - \mathrm{negl}(\lambda) \ .$$

Using the above recovery guarantee, we lower bound the success probability of Inv as follows.

$$\begin{aligned} &\Pr\left[\begin{array}{rl} g & \leftarrow \mathsf{Sample}(1^\lambda) \\ x & \leftarrow \{0,1\}^n \end{array} : \mathsf{Inv}(g, g(x)) = x \right] \\ &\geq \Pr\left[\begin{array}{rl} g & \leftarrow \mathsf{Sample}(1^\lambda) \\ (x,x') & \leftarrow \{0,1\}^{2n} \end{array} : \mathsf{Rec}(g, g(x), x') = x \,\middle|\, A \right] \cdot \mathcal{U}(A) \\ &\geq (1 - \mathrm{negl}(\lambda)) \cdot \mathcal{U}(A) \ . \end{aligned}$$

But from the earlier information-theoretic bound, no function g with output size k can support inversion with success probability greater than $2^{-(n-k)}$. Hence, we conclude

$$2^{-(n-k)} \geq \mathcal{U}(A) \cdot (1 - \mathrm{negl}(\lambda)) \ ,$$

For sufficiently large λ, we have $\mathrm{negl}(\lambda) < 1/2$, which implies

$$k \geq n + \log \mathcal{U}(A) \ .$$

□

Claim B.6. *Let $r = r(n)$ be any efficiently computable function. Then, the uniform distribution $\mathcal{U}$ over pairs $(x, x') \in \{0,1\}^n \times \{0,1\}^n$ conditioned on $\|x - x'\|_0 = r$ is efficiently sampleable.*

Proof. First, sample $x \leftarrow \{0,1\}^n$ uniformly at random. Then sample an r-sparse perturbation vector $\Delta \in \{0,1\}^n$ independently of x. Finally, set $x' = x \oplus \Delta$. □

Corollary B.7. *Let $r = r(n)$ be any efficiently computable function. Then, for all sufficiently large n, the sketch size of any sharp sketching scheme for $\mathsf{Ham}_{r/n}$ is at least $\log \binom{n}{r}$.*