Inaccessible Entropy for Watermarking Generative Agents

Daniel Alabi*[1] and Lav R. Varshney†[2]

[1,2]University of Illinois Urbana-Champaign
[1]Columbia University

Abstract

In this work, we construct *distortion-free* and *unforgeable* watermarks for language models and generative agents. The watermarked output cannot be forged by a adversary nor removed by the adversary without significantly degrading model output quality. That is, the watermarked output is distortion-free: the watermarking algorithm does not noticeably change the quality of the model output and without the public detection key, no efficient adversary can distinguish output that is watermarked from outputs which are not. The core of the watermarking schemes involve embedding a message and publicly-verifiable digital signature in the generated model output. The message and signature can be extracted during the detection phase and verified by any authorized entity that has a public key. We show that, assuming the standard cryptographic assumption of one-way functions, we can construct *distortion-free* and *unforgeable* watermark schemes. Our framework relies on analyzing the inaccessible entropy of the watermarking schemes based on computational entropy notions derived from the existence of one-way functions.

*alabid@illinois.edu
†varshney@illinois.edu

1 Introduction

Generative agents powered by advances in artificial intelligence (AI) and machine learning are revolutionizing fields ranging from creative content generation to autonomous decision-making and scientific inquiry [BMR+20]. However, their widespread use has also introduced significant concerns on the misuse of generated content, intellectual property rights, and authenticity of digital assets. In this context, watermarking has emerged as a critical tool for ensuring traceability, authenticity, and security in generative models [LMC+24], and is already seeing large-scale deployment [DSG+24]. Traditional watermarking methods in cryptography, while effective in many applications, cannot be seamlessly integrated into generative agents [KJGR21].

At the same time, there has been a large body of work on detecting AI-generated content from specific language models [GPT23, GSR19, Ber16, Hov16, DMFZ22]. The problem most detectors tackle can be framed as follows: *given some generated content from a language model, can you detect if it was generated by a human or a specific language model?* However, such detection schemes have a high false-positive or false-negative rates, especially as models get better in output quality [KSK+23, ZHR+19, JAML20, TCH23, CBZ+23, VKS20]. This difficulty necessitates watermarking.

Instead of relying on a trained classifier to distinguish machine-generated content from human-generated content, a recent line of work modifies the language model generation process itself to introduce watermarks [Aar23, KGW+23, KTHL24]. The generation process intentionally makes the watermarked output distribution different from its non-watermarked output distribution. During the detection phase, as long as the two distributions are far enough apart, the detector can distinguish watermarked text from non-watermarked text for a particular language model. In particular, Christ et al. [CGZ23] take a cryptographic approach to develop distortion-free watermarks for language models, assuming the minimal assumption of existence of one-way functions. Their watermark provides strong theoretical guarantees of soundness, completeness, and distortion-freeness (i.e., the watermarking process does not noticeably alter the output distribution). However, their watermarks are neither publicly-verifiable nor provide unforgeability guarantees. Also, they assume there is a shared key between the model provider and the detector.

We present *distortion-free* and *unforgeable* watermarks for generative models and language models with theoretical guarantees of soundness and completeness (and relying on minimal cryptographic assumptions without use of random oracles [GHR99]). Our work introduces a novel framework—based on notions of inaccessible entropy—that can be tailored to generative models. The watermarking scheme can be seamlessly integrated on top of existing generative models and are:

- **Distortion-free**: The difference between watermarked model outputs and non-watermarked model outputs are imperceptible to adversaries (that do not have access to a public key) because the quality of the generated content does not degrade.
- **Unforgeable**: The cryptographic techniques ensure that only authorized entities can create or validate the embedded watermarks. This prevents unauthorized replication or tampering from adversaries who do not have access to the secret key. The model outputs can be verified using a public key. In addition, the model designer cannot claim that the watermark does not exist once it has been embedded in the output.

In other words, the property of *distortion-freeness* ensures watermarks do not compromise the quality or functionality of generated content, whereas *unforgeability* guarantees that only authorized entities can embed these watermarks. Achieving both goals simultaneously in generative models has been a significant challenge. Recent work [LPH+24] presents unforgeable watermarks but does not come with theoretical security guarantees. Fairoze et al. [FGJ+23] present distortion-free and unforgeable watermarks with formal guarantees of soundness and completeness. However, they rely on random oracles, which are impossible to implement in practice [MMP14].

Our work can be seen as a way to remove the random oracle assumptions from previous watermarking schemes via the lens of inaccessible entropy. We show soundness for the distortion-free and unforgeability properties of watermarking schemes. That is, if the model output is not generated without knowledge of the secret key, then the detector will not state otherwise (except with negligible probability). Unforgeability protects language model designers and users. Once the model user generates content, they cannot remove or alter the watermark associated with the content. This protects downstream content consumers from potentially malicious model providers who might, in the future, claim that they did not generate some certain content. Also, unforgeability protects model providers so that content consumers of their model cannot claim copyright over content that traces back to the original model.

Next, we present the main results and describe their implications.

1.1 Main Results

We will prove the following theorem.

Theorem 1.1 (Informal version of Theorem 5.5)**.** *Suppose there exists a one-way function. Then there exists a distortion-free and unforgeable publicly-detectable watermarking scheme. Furthermore, the scheme satisfies the properties of (weak) robustness, soundness, and completeness.*

Theorem 1.1 can be shown via the construction of a publicly-detectable watermarking scheme. The desired desiderata are distortion-freeness, weak robustness, soundness, and completeness. We show that the watermarking scheme satisfies all these properties (assuming one-way permutations).

The scheme relies on a new construction of error-correcting codes with the primary goals of ensuring: (1) The message-signature pair can be encoded and that, given the message, the signature can be decoded; (2) The probability of finding message-signature pairs that result in a codeword collision is negligible (with respect to the security parameter).

Theorem 1.2 (Informal version of Theorem 5.2)**.** *Let λ be the security parameter. Also, let $\boldsymbol{\sigma}$ be a signature, $\mathbf{m}$ be a message, and $\mathbf{c}$ be a codeword. Then* $(\mathsf{HashEncode}, \mathsf{HashDecode})$ *are functions such that* $\mathsf{HashDecode}(\mathsf{HashEncode}(\mathbf{m}, \boldsymbol{\sigma}), \mathbf{m}) = \boldsymbol{\sigma}$ *with probability at least* $1 - \mathrm{neg}(\lambda)$*. And for every codeword $\mathbf{c}$ and message-signature pair $(\mathbf{m}, \boldsymbol{\sigma})$ such that $\mathbf{c} \leftarrow \mathsf{HashEncode}(\mathbf{m}, \boldsymbol{\sigma})$, only with probability at most* $\mathrm{neg}(\lambda)$ *can a polynomial-time adversary find $(\mathbf{m}', \boldsymbol{\sigma}')$ such that $(\mathbf{m}, \boldsymbol{\sigma}) \neq (\mathbf{m}', \boldsymbol{\sigma}')$ and $\mathbf{c} = \mathsf{HashEncode}(\mathbf{m}, \boldsymbol{\sigma}) = \mathsf{HashEncode}(\mathbf{m}', \boldsymbol{\sigma}')$.*

Suppose there exists a one-way function $f : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$. Then there exists a family of $(\mathsf{HashEncode}, \mathsf{HashDecode})$ *functions with codeword length $\widetilde{O}(\lambda^7)$.*

We define the properties of the $(\mathsf{HashEncode}, \mathsf{HashDecode})$ functions in Definition 5.1. The $(\mathsf{HashEncode}, \mathsf{HashDecode})$ functions are used in the generation and detection phases of the watermarking scheme. Theorem 1.2 is agnostic to some specific properties of the error-correcting code (e.g., the distance). The better the distance of the code, the better the construction of $(\mathsf{HashEncode}, \mathsf{HashDecode})$.

1.2 Comparison to Previous Works on Watermarking Generative Agents

Our scheme achieves a form of (weak) robustness: the output of the generative model can be modified (up to some threshold) and the detector will still detect the presence of a watermark. In the general case, Zhang et al. [ZEF+] show that it is impossible to achieve strong robustness (i.e., the watermarked output is still detectable after substantially changing its content). The attack by Zhang et al. [ZEF+] works against shared-key watermarking schemes [CG24, GM24, GG24, AAC+24, ZALW24, ZWL24, ZGC+24, Aar23, KTHL24]. Since our scheme only achieves weak robustness, their attack is not applicable to our setting.

No previous work attains the same (or better) security and robustness properties as our watermarking scheme while relying on the same (or weaker) assumptions.

Result	Distortion-Free?	Weak Robustness?	Unforgeable?	No Random Oracles?
[Aar22]	NO	NO	NO	YES
[KGW+23]	NO	YES	NO	YES
[CGZ23]	YES	YES	NO	YES
[FGJ+25]	YES	YES	YES	NO
This work	YES	YES	YES	YES

Table 1: Comparing watermarking schemes for language models. In this table, weak robustness means that the watermark is resilient to a few edits. The distortion-free property means that answers to a query to the watermarked model are computationally indistinguishable from those of the non-watermarked model.

Recent work on pseudorandom codes derives constructions of codes that satisfy certain definitions of pseudorandomness and that are robust to edits in the model output. In particular, Christ and Gunn [CG24] constructed pseudorandom error-correcting codes to tolerate a constant rate of errors. They apply their codes to watermarking and to steganography. There have been follow-ups on those codes [GM24, GG24]. Notably, their constructions are based on the hardness of LPN (Learning Parity with Noise), the planted XOR problem at low density, or similar assumptions [BFKL93, BKW03]. These assumptions are different (and arguably stronger) than those in our work [BLVW19]: we aim to understand what formal guarantees we can achieve via the use of one-way functions or permutations. Furthermore, our work is not focused on constructing new codes with security parameters (i.e., code-based cryptography [EOS06]) but on using existing non-cryptographic codes to construct primitives with security properties. On the other hand, the previous work on pseudorandom codes seem to result in constant-rate codes [CG24] whereas Definition 1.2 does not result in a constant-rate encoding of the signature.

Another important distinction between our work and previous works is that we model the interaction between generative agents in accessible entropy and make no assumption about whether the receiver of the model output is human or another agent.

1.3 Overview of Techniques

We provide a brief overview of our modeling approach and techniques.

1.3.1 Generative Agent as Noisy Channel

We can model token generation in the generative model (beyond auto-regressive models) as a noisy channel. The noisy channel model assumes that there is an original message that passes through a noisy channel, which introduces distortions or uncertainties, leading to a potentially different observed message. The goal is to reconstruct the most likely original message given the noisy output. In our work, the noisy channel is the scheme used to generate the tokens. In particular, both the watermarked and non-watermarked generative model can be modeled as a noisy channel which allows for analyses of the flow of information through the channel [Sha49a, Sha49b].

Using the noisy channel approach, we can improve token generation by: (1) *Re-ranking token candidates*: Instead of taking the highest probability token at each step, we can use a noisy channel correction model to re-rank generated outputs. This is, essentially, what the watermarking algorithm accomplishes in Watermark (Algorithm 4). The channel is able to embed a signal in the output that allows for detecting the presence of a watermark. (2) *Applying error correction*: By estimating the probability that a given token was generated due to controlled noise (i.e., due to watermarking), we can decide if the given output is watermarked or not. However, the amount of controlled noise introduced for watermarking might be excessive. As discussed in previous works, this situation corresponds to low-entropy responses to prompts [KGW+23, CGZ23]. Thus, we apply error-correction on the noisy channel. As we will show, the number of errors that the error-correcting codes can handle is at most the number of low-entropy periods the watermarking scheme can tolerate (Lemma 5.3).

1.3.2 Inaccessible Entropy in Error-Correcting Codes

Real and accessible entropy is defined and used in previous work of Haitner et al. to construct statistically hiding commitments and to give statistical zero-knowledge arguments for any NP statement. They defined computational notions of entropy (i.e., accessible and inaccessible entropy) to measure the infeasibility of sampling high entropy strings that are consistent with a cryptographic protocol [HNO+09, HRVW09, HHR+10, HV17].

In our work, the cryptographic protocol is the watermarking scheme and we analyze the scheme via the lens of computational entropy notions. Specifically, we use accessible and inaccessible entropy notions to analyze the construction of the family of (HashEncode, HashDecode) functions which is used in the watermarking schemes. These functions are used to enforce computational indistinguishability of the generative model outputs. See Definition 5.1 that defines properties that (HashEncode, HashDecode) should have. See Theorem 5.1 that instantiates the inaccessible entropy framework to construct the family of (HashEncode, HashDecode) functions. All error-correcting codes are defined by encoding and decoding functions that map messages to codewords and vice versa. In the watermarking scheme, the functions (HashEncode, HashDecode) are used to encode the embedded signature and to recover the message during the detection phase.

1.3.3 Entropy Manipulations

We also rely on previous techniques for manipulation and quantification of real and accessible entropy. Specifically, the construction of (HashEncode, HashDecode) functions applies some transformations to obtain noticeable gaps between the real Shannon entropy and accessible average max-entropy.

The transformations can be summarized as follows: the gap amplification operation via a direct product increases the gap between the real and accessible entropy; the entropy reduction operation uses hashing on the inputs to reduce the accessible entropy of a function while (approximately) preserving the gap between the real min-entropy and accessible max-entropy; then by hashing outputs, the output length of the constructed function can be reduced. See Theorem 5.2 for more details.

1.4 Organization of Subsequent Sections

Section 2 discussed some core related works in the information-hiding literature, connections to digital signatures, and software watermarking. Section 3 sets up the rest of the paper by standardizing notation, providing definitions for accessible and inaccessible entropy, and giving the basics of coding theory. Section 4 discusses the security model under which our watermarking schemes operate. Section 5 defines and analyzes the constructions used to achieve the publicly-detectable watermarking scheme.

2 Other Related Work

2.1 Information Hiding

Moulin and O'Sullivan [MO03] survey the information-theoretic foundations of *information hiding* for pre-generated signals, which includes watermarking as a special case. The problem of information hiding can be framed as a strategic interaction between two opposing teams—the embedder (and decoder) and the attacker. Modeling information hiding allows for analysis of optimal information embedding and attack strategies. The information-theoretic and associated coding-theoretic framework [MO03, MK05] also yields approaches for designing near-optimal codes and universal decoders for information hiding. Theoretical insights are illustrated through applications to image watermarking and attacks on watermarked images [Mou03, Mou05]. Geometric attacks are a popular class of attacks in the image watermarking literature, where a warping operation is applied to an image.

This previous work focuses on identifying the fundamental limits of how much hidden information can be embedded into host signals (such as images, audio, or video) without being detected by unauthorized parties or without incurring unacceptable distortion. By using tools from rate-distortion theory, channel coding, and hypothesis testing, theoretical frameworks relate embedding rate (how much data is hidden), embedding fidelity (how much distortion is introduced in the host signal), and detectability (the probability that an adversary can detect and/or remove the hidden signal). Recent work by He et al. revisits such basic information-theoretic trade-offs in the context of dynamically-generated content (as in generative agents), rather than pre-generated signals [HLW+24, HLW+25].

In other recent work, Abdelnabi and Fritz [AF21] address the challenge of tracing the origin of text generated by language models. The authors propose the adversarial watermarking transformer (AWT), an end-to-end model that unobtrusively encodes binary messages into input text. AWT employs a jointly trained encoder-decoder architecture with adversarial training to ensure minimal alterations to the original text's semantics and correctness. The model automatically learns optimal word substitutions and their placements without requiring ground truth data. This framework can

be seen as an instance of information hiding where AWT acts as the embedder (and decoder) and the adversarial trainer models the attacker.

In many watermarking applications, the adversary's goal is to detect or estimate the hidden information, so a natural question is: *Under what conditions (and at what rates) can a watermark be statistically indistinguishable from noise?* In our work, we additionally consider computational indistinguishability notions not typically considered in the information-theoretic information hiding literature.

2.2 Software Watermarking

A closely related area to watermarking of generative model outputs is *software watermarking*. The concept of software watermarking involves embedding unique identifiers into programs to assert ownership and prevent unauthorized use or distribution [CT99].

Traditional approaches to software watermarking have faced challenges, particularly concerning the robustness of the watermark against removal and the ability to publicly verify the watermark's presence. In their seminal work, Barak et al. [BGI+12] introduced the notion of cryptographic software watermarking and demonstrated that the existence of indistinguishability obfuscation implies the impossibility of certain forms of watermarking. This result suggested limitations in creating watermarks that are both resilient and publicly verifiable. Subsequent research explored various methods to overcome some of these impossibility results. Naccache, Shamir, and Stern [NSS99] and Nishimaki [Nis13] proposed software watermarking schemes that offered only secret key verification (rather than public key verification) and was susceptible to specific types of attacks. Cohen, Holmgren, and Vaikuntanathan [CHV15] introduced a publicly-verifiable watermarking scheme for families of puncturable pseudorandom functions (PPRF), leveraging indistinguishability obfuscation and injective one-way functions. Instead of requiring the watermarked program to agree with the original unmarked program on all inputs (to the program), it is only required that they agree on a large fraction of inputs.

There is a large literature on watermarking of software and programs [CT99, BGI+12, NSS99, Nis13, CHV15]. However, most of these programs cannot be readily applied to generative language models without significant modifications to the weights of the trained language model (an expensive process!). In our work, we do not require modification of model weights.

2.3 Code-Based Cryptography

Code-based cryptography is the study of cryptographic systems in which the security relies on hardness of solving coding-theoretic problems. This field was pioneered by McEliece [McE78] and Niederreiter [Nie86, Sen11] decades ago. The McEliece cryptosystem is based on the difficulty of decoding general linear codes, has withstood decades of cryptanalysis and remains a strong candidate for post-quantum cryptography [MTSB13]. The cryptosystem is one of the earliest public-key encryption schemes based on error-correcting codes [EOS06]. It leverages the hardness of syndrome decoding in general linear codes, particularly Goppa codes [Ber73].

There is, by now, a large literature on code-based cryptography. Recent work on pseudorandom error-correcting codes [CG24] show applications of code-based cryptography to watermarking. They design new codes with pseudorandom and robustness properties. In our work, we focus on modifying existing codes (or incorporating non-cryptographic codes) to achieve security guarantees rather than creating new code constructions.

2.4 Digital Signatures

Digital signatures have been extensively studied as a fundamental tool for ensuring authenticity, integrity, and non-repudiation in digital communication. The development of digital signatures can be traced back to Diffie and Hellman's pioneering work on public-key cryptography [DH76], which laid the foundation for modern cryptographic protocols. Since then, numerous schemes have been proposed, each with varying security guarantees and efficiency considerations [RSA78, Lam79, BG89, Mer87, GMR88, EGM89, BG89].

The earliest and most widely used digital signature schemes are based on number-theoretic problems, including the Discrete Logarithm Problem and the Integer Factorization Problem [RSA78, Lam79, BG89, Mer87]. Some of the most significant works in this area include the work by Rivest, Shamir, and Adleman that introduces the RSA Digital Signature Algorithm [RSA78, DHT12]. ElGamal proposed a signature scheme based on the discrete logarithm problem, later leading to the development of the digital signature algorithm [ElG84]. In addition, there is work on post-quantum digital signatures that, generally, do not rely on hardness of discrete logarithm and factorization problems [TSZ19]. Our work does not concern the development of new digital signature schemes but use of existing signature schemes for watermarking schemes.

Our watermarking scheme cannot directly use previous works on digital signatures as the message-signature pair would make the resulting model output no longer "distortion-free." Our scheme uses hashing to embed the signature into the language model output.

2.5 Generative Agents

In classical cryptography, security guarantees are with respect to users that are usually assumed to be human. Here, we make no distinction in providing security guarantees for humans versus generative agents. Generative agents are autonomous systems powered by generative artificial intelligence (AI) models, designed to create, adapt, and respond in various contexts by producing new and contextually relevant outputs. These agents leverage underlying generative technologies—such as language models, image generation models, or multi-modal AI systems—to perform tasks, engage in interactions, and learn from their environments. Examples of generative agents include chatbots and online virtual characters. Generative agents are already deployed for use in healthcare, education, and customer support [POC+23].

3 Preliminaries

3.1 Notation

We use calligraphy to denote sets, lowercase for values, uppercase for random variables, bold face for vectors, and sans serif for algorithms (i.e., Turing machines). For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. For vector $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathcal{J} \subseteq [n]$, let $\mathbf{y}_{\mathcal{J}} = (y_{i_1}, \ldots, y_{i_{|\mathcal{J}|}})$, where $i_1 < \cdots < i_{|\mathcal{J}|}$ are the elements of $\mathcal{J}$. Let a_i denote the ith bit of vector $\mathbf{a}$. For clarify, we also use slicing notation: $\mathbf{a}_{[-i]}$ refers to the ith last element of a list and $\mathbf{a}_{[j:k]}$ extracts the elements a_i for $i \in [j, k)$. Let $a \parallel b$ denote the concatenation of a to b. We use $\log(\cdot)$ to take logarithms base 2. Let () denote an empty list or empty string. Also, let PPT denote a probabilistic polynomial-time algorithm and let poly denote the set of all positive polynomials. The operation $\oplus$ corresponds to the simple XOR that

is applied bit by bit. That is, for any $\mathbf{a}, \mathbf{b}$ of length n, for any $i \in [n]$, $(\mathbf{a} \oplus \mathbf{b})_i = \mathbf{a}_i \oplus \mathbf{b}_i$ where $\mathbf{a}_i \oplus \mathbf{b}_i$ is the bit XOR of $\mathbf{a}_i$ and $\mathbf{b}_i$.

For the cryptographic primitives in this paper, we use λ for the security parameter. A negligible function $\mathrm{neg}(\lambda)$ in λ are those functions that decay faster than the inverse of any polynomials. That is, for all $\mathrm{poly}(\lambda)$, it holds that $\mathrm{neg}(\lambda) < \frac{1}{\mathrm{poly}(\lambda)}$ for all large enough λ.

Definition 3.1 (Negligible Function)**.** *A function $\nu\colon \mathbb{N} \mapsto [0,1]$ is negligible, denoted $\nu(\lambda) = \mathrm{neg}(\lambda)$, if $\nu(\lambda) < 1/p(\lambda)$ for every $p \in \mathrm{poly}$ and large enough λ.*

We use $\xleftarrow{R}$ to denote a random sample, e.g., $r \xleftarrow{R} \{0,1\}^n$ to sample n random bits. We use an asterisk to denote an arbitrary-length string of tokens from a set of possible tokens, e.g., $\mathcal{S}^*$ for a given set $\mathcal{S}$. Often, we use U_n to represent the uniform distribution supported on $\{0,1\}^n$ and $\xleftarrow{R} U_n$ to denote a random sample from the distribution, e.g., $r \xleftarrow{R} U_n$.

For an event E, $\mathbb{P}[E]$ is the probability of event E occurring. Let X and Y be random variables taking values in a discrete universe $\mathcal{U}$. We adopt the convention that, when the same random variable appears multiple times in a specific expression, all occurrences refer to the same instantiation. For example, $\mathbb{P}[Y = Y]$ is 1. For an event E, we write $X|_E$ to denote the random variable X conditioned on E. We let $\mathbb{P}_{X|Y}[x|y]$ stand for $\mathbb{P}[X = x \mid Y = y]$. The *support* of a random variable X, denoted $\mathrm{Supp}(X)$, is defined as $\{x\colon \mathbb{P}[X = x] > 0\}$. Let U_n denote a random variable that is uniform over $\{0,1\}^{n(\lambda)}$. For $t \in \mathbb{N}$, let $X^{(t)} = (X^1, \ldots, X^t)$, where $X^1, \ldots, X^t$ are independent copies of X. We write $X \equiv Y$ to indicate that X and Y are identically distributed.

Definition 3.2 (Statistical Difference)**.** *We write $\mathrm{SD}(X,Y)$ to denote the* statistical difference *(also known as variational distance) between X and Y, i.e.,*

$$\mathrm{SD}(X,Y) := \max_{T \subseteq \mathcal{U}} |\mathbb{P}[X \in T] - \mathbb{P}[Y \in T]|.$$

If $\mathrm{SD}(X,Y) \le \varepsilon$ [resp., $\mathrm{SD}(X,Y) > \varepsilon$], we say that X and Y are *ε-close* [resp., ε-far].

Indistinguishability Two random variables $X = X(\lambda)$ and $Y = Y(\lambda)$ are *statistically indistinguishable*, denoted $X \approx_s Y$, if for any unbounded algorithm D, it holds that

$$\Big|\mathbb{P}[\mathsf{D}(1^\lambda, X(\lambda)) = 1] - \mathbb{P}[\mathsf{D}(1^\lambda, Y(\lambda)) = 1]\Big| = \mathrm{neg}(\lambda).$$

Note that this is equivalent to requiring that $\mathrm{SD}(X(\lambda), Y(\lambda)) = \mathrm{neg}(\lambda)$.

Similarly, X and Y are *computationally indistinguishable*, denoted $X \approx_c Y$, if

$$\Big|\mathbb{P}[\mathsf{D}(1^\lambda, X(\lambda)) = 1] - \mathbb{P}[\mathsf{D}(1^\lambda, Y(\lambda)) = 1]\Big| = \mathrm{neg}(\lambda),$$

for every PPT D.

3.2 Digital Signatures

A digital signature is a scheme used to verify the authenticity of digital documents or messages [DH76]. The recipient of a signed message should be able to confidently verify that the

digital signature has not been forged by an adversary and that the message came from the purported sender [KL14]. It is known that signature schemes can be constructed from one-way functions [Lam79, Rom90].

For generative models, we will embed the digital signature directly into the model output. We rely on a public-key signature scheme with the following properties.

Definition 3.3 (Public-Key Signature Scheme)**.** *A public-key signature scheme* S *is a tuple of algorithms* $\mathsf{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *where:*

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$ *outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$ *with respect to the security parameter* λ*, where* $(\mathsf{sk}, \mathsf{pk})$ *are the private key and public key, respectively.*
- $\mathsf{Sign}_{\mathsf{sk}}(m) \to \boldsymbol{\sigma}$ *produces a signature* $\boldsymbol{\sigma}$*, given a message* m*, using the secret signing key* sk*. We denote the signature size* $|\boldsymbol{\sigma}|$ *by* $\lambda_{\boldsymbol{\sigma}}$*.*
- $\mathsf{Verify}_{\mathsf{pk}}(m, \boldsymbol{\sigma}) \to \{\mathtt{true}, \mathtt{false}\}$ *outputs* `true` *or* `false`*, given a candidate message* m *and signature* $\boldsymbol{\sigma}$*, using the public verification key.* $\mathsf{Verify}_{\mathsf{pk}}$ *is a deterministic verification algorithm that outputs* `true` *if and only if the message-signature pair is valid.*

Except with negligible probability over $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{Gen}(1^\lambda)$, we require that

$$\mathsf{Verify}_{\mathsf{pk}}(m, \mathsf{Sign}_{\mathsf{sk}}(m)) = 1,$$

for every legal message m. We call $\boldsymbol{\sigma}$ a *valid signature* on a message m if $\mathsf{Verify}_{\mathsf{pk}}(m, \boldsymbol{\sigma}) = \mathtt{true}$.

Definition 3.4 (Unforgeability)**.** *For every adversary* D*, we have*

$$\mathbb{P}\left[\mathsf{Verify}_{\mathsf{pk}}(m^*, \boldsymbol{\sigma}^*) = \mathtt{true} \quad : \quad \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m^*, \boldsymbol{\sigma}^*) \leftarrow \mathsf{D}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk}) \end{array} \right] \leq \mathrm{neg}(\lambda).$$

Here, the adversary gets oracle access to the signing oracle $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$*, but* m^* *in the final forgery output* $(m^*, \boldsymbol{\sigma}^*)$ *must have never been queried using the signing oracle. As a signature scheme, we will require this property to guarantee that it is hard to forge a watermark.*

3.2.1 Hash-and-Sign Paradigm

The Hash-and-Sign paradigm is an approach in the use of digital signature schemes that allows for the signing of arbitrarily long model outputs.

It involves two main steps:

- *Hashing*: Before signing, the message is first fed to a (target) collision resistant hash function to produce a fixed-size digest.
- *Signing*: The signer then applies a digital signature algorithm to the hash, rather than the full message, to generate the signature.

As long as we have (target) collision-resistant hash functions, we can construct digital signatures [GHR99].

Theorem 3.5 (Theorem 12.4 in [KL14])**.** *If* Π *is a secure signature scheme for messages of length* ℓ *and* Π_H *is target collision-resistant, then the construction in Algorithm 1 is a secure (unforgeable) signature scheme for arbitrary-length messages, satisfying Definition 3.4.*

Algorithm 1 HashAndSign Paradigm

1: Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme for messages of length $\ell(\lambda)$.
2: Let $\Pi_H = (\mathsf{Gen}_H, H)$ be a hash function with output length $\ell(\lambda)$.
3: Construct signature scheme $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$ as follows:

- Gen': on input 1^λ:
  - Run $\mathsf{Gen}(1^\lambda)$ to obtain $(\mathsf{sk}, \mathsf{pk})$
  - Run $\mathsf{Gen}_H(1^\lambda)$ to obtain r
  - The public key is $(\mathsf{pk}, \mathsf{r})$ and the private key is $(\mathsf{sk}, \mathsf{r})$
- Sign': On input a private key $(\mathsf{sk}, \mathsf{r})$ and message $m \in \{0,1\}^*$, output $\mathsf{Sign}_{\mathsf{sk}}(H(\mathsf{r} \parallel m)) \to \boldsymbol{\sigma}$
- Verify': On input a public key $(\mathsf{pk}, \mathsf{r})$, a message $m \in \{0,1\}^*$, and a signature $\boldsymbol{\sigma}$, output `true` if and only if $\mathsf{Verify}_{\mathsf{pk}}(H(\mathsf{r} \parallel m), \boldsymbol{\sigma})$ is `true`.

3.3 Entropy Definitions

Throughout this paper, we refer to and use several measures of entropy. The entropy definitions can be stated in terms of the sample entropy.

Definition 3.6 (Sample Entropy)**.** *For a random variable X and $x \in \mathrm{Supp}(X)$, the* sample entropy *of x with respect to X is the quantity*

$$\mathrm{H}_{\mathrm{X}}(x) := \log \tfrac{1}{\mathbb{P}[X=x]},$$

letting $\mathrm{H}_{\mathrm{X}}(x) = \infty$ for $x \notin \mathrm{Supp}(X)$, and $2^{-\infty} = 0$.

For the specific sample x, the sample entropy measures the amount of "randomness" in x, assuming that x has been generated according to X. Using this notion, we can define the *Shannon entropy* $\mathrm{H}(X)$, *min-entropy* $\mathrm{H}_\infty(X)$, and *max-entropy* as follows.

Definition 3.7 (Shannon Entropy)**.**

$$\mathrm{H}(X) := \mathop{\mathbb{E}}_{x \xleftarrow{R} X} [\mathrm{H}_{\mathrm{X}}(x)].$$

Definition 3.8 (Min-Entropy)**.**

$$\mathrm{H}_\infty(X) := \min_{x \in \mathrm{Supp}(X)} \mathrm{H}_{\mathrm{X}}(x).$$

Definition 3.9 (Max-Entropy)**.**

$$\mathrm{H}_0(X) := \log |\,\mathrm{Supp}(X)|.$$

Definition 3.10 (Collision Probability)**.** *The* collision probability *of X is*

$$\mathrm{CP}(X) := \sum_{x \in \mathrm{Supp}(X)} \mathbb{P}_X[x]^2 = \mathbb{P}_{(x,x') \xleftarrow{R} X^2}\big[x = x'\big].$$

This leads to a corresponding *Rényi entropy* as follows.

Definition 3.11 (Rényi Entropy)**.**

$$\mathrm{H}_2(X) := -\log \mathrm{CP}(X).$$

It can be shown that

$$\mathrm{H}_\infty(X) \le \mathrm{H}_2(X) \le \mathrm{H}(X) \le \mathrm{H}_0(X).$$

Further, $\mathrm{H}_\infty(X) = \mathrm{H}_2(X) = \mathrm{H}(X) = \mathrm{H}_0(X)$ if and only if X is flat (uniform on its support).

In general, stating that $\mathrm{H}_\infty(X) \ge k$ is a strong way of saying that X has "high entropy" and $\mathrm{H}_0(X) \le k$ a strong way of saying that X has "low entropy".

We will also be interested in conditional versions of entropy.

Definition 3.12 (Conditional Entropy)**.** *For jointly distributed random variables* (X, Y) *and* $(x, y) \in \mathrm{Supp}(X, Y)$*, the* conditional sample-entropy *is* $\mathrm{H}_{X|Y}(x|y) = \log \frac{1}{\mathbb{P}_{X|Y}[x|y]} = \log \frac{1}{\mathbb{P}[X=x|Y=y]}$*. Then the standard* conditional Shannon entropy *can be written as*

$$\mathrm{H}(X \mid Y) = \mathop{\mathbb{E}}_{(x,y) \xleftarrow{R} (X,Y)} \left[\mathop{\mathrm{H}}_{X|Y}(x \mid y) \right] = \mathop{\mathbb{E}}_{y \xleftarrow{R} Y} \left[\mathrm{H}(X|_{Y=y}) \right] = \mathrm{H}(X, Y) - \mathrm{H}(Y).$$

3.4 Inaccessible Entropies

In this section, we recall definitions of real and accessible entropy from [HNO+09, HRVW09, HHR+10, HV17].

3.4.1 Real Entropy

For a computable function F, we define the *real entropy* of F^{-1} to be the amount of entropy left in the input after revealing the output. We measure the entropy using Shannon entropy (average case), min-entropy, and max-entropy.

Let $\mathsf{F} : \{0,1\}^{n(\lambda)} \to \{0,1\}^m$ and let $\mathsf{F}^{-1} : \{0,1\}^m \to (\{0,1\}^{n(\lambda)})^*$ where we define

$$\mathsf{F}^{-1}(y) = \{x \,:\, F(x) = y\},$$

and

$$\mathsf{F}^{-1}(\mathcal{Y}) = \{x \,:\, \exists y \in \mathcal{Y}, x \in \mathsf{F}^{-1}(y)\}.$$

Definition 3.13 (Real Entropy)**.** *Let* $n = n(\lambda)$ *be a security parameter, and* $\mathsf{F} \colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$ *a function. We say that* F^{-1} *has* real Shannon entropy k *if*

$$\mathrm{H}(X \mid \mathsf{F}(X)) = k,$$

where X *is uniformly distributed on* $\{0,1\}^{n(\lambda)}$*. We say that* F^{-1} *has* real min-entropy *at least* k *if there is a negligible function* $\mathrm{neg} = \mathrm{neg}(n(\lambda))$ *such that*

$$\mathbb{P}_{x \xleftarrow{R} X} \left[\mathop{\mathrm{H}}_{X|\mathsf{F}(X)}(x \mid \mathsf{F}(x)) \ge k \right] \ge 1 - \mathrm{neg}(n(\lambda)).$$

We say that F^{-1} *has* real max-entropy *at most* k *if there is a negligible function* $\mathrm{neg} = \mathrm{neg}(n(\lambda))$ *such that*

$$\mathbb{P}_{x \xleftarrow{R} X} \left[\mathop{\mathrm{H}}_{X|\mathsf{F}(X)}(x \mid \mathsf{F}(x)) \le k \right] \ge 1 - \mathrm{neg}(n(\lambda)).$$

It is easy to verify that, ignoring negligible terms, the min-entropy of F^{-1} is at most its Shannon entropy, which in turn is at most its max-entropy, where equality holds only if F is regular.

We can construct *universal one-way hash functions* (UOWHFs) that are shrinking, achieving high real entropy as a natural intermediate step. Indeed, the amount by which F shrinks is a lower bound on the real entropy of F^{-1}.

Proposition 3.14 (See [HHR+20])**.** *If* $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$*, then the real Shannon entropy of* F^{-1} *is at least* $n-m$*, and the real min-entropy of* F^{-1} *is at least* $n-m-s$ *for any* $s=\omega(\log n)$*.*

Proof. For Shannon entropy, we have

$$\mathrm{H}(X \mid \mathsf{F}(X)) \geq \mathrm{H}(X) - \mathrm{H}(\mathsf{F}(X)) \geq n-m.$$

For min-entropy, let $S = \{y \in \{0,1\}^m \colon \mathbb{P}[\mathsf{F}(X)=y] < 2^{-m-s}\}$. Then $\mathbb{P}[\mathsf{F}(X) \in S] \leq 2^m \cdot 2^{-m-s} = \mathrm{neg}(n(\lambda))$, and for every x such that $F(x) \notin S$, we have

$$\underset{X|\mathsf{F}(X)}{\mathrm{H}}(x \mid \mathsf{F}(x)) = \log \frac{1}{\mathbb{P}[X=x \mid \mathsf{F}(X)=\mathsf{F}(x)]} = \log \frac{\mathbb{P}[\mathsf{F}(X)=\mathsf{F}(x)]}{\mathbb{P}[X=x]} \geq \log \frac{2^{-m-s}}{2^{-n}} = n-m-s.$$

□

3.4.2 Accessible Entropy

We define accessible entropy of F^{-1} using the notion of "collision-finding" algorithm, an algorithm that aims to find a second-pre-image of $\mathsf{F}(X)$ with "maximal entropy". The accessible entropy of F will be defined as the entropy of the best *efficient* collision-finding algorithm.

Definition 3.15 (Collision Finding Algorithm)**.** *For a function* $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$*, an* F-collision-finder *is a randomized algorithm* A *such that for every* $x \in \{0,1\}^{n(\lambda)}$ *and coin tosses* r *for* A*, we have* $\mathsf{A}(x;r) \in \mathsf{F}^{-1}(\mathsf{F}(x))$*.*

Note that A is required to *always* produce an input $x' \in \{0,1\}^{n(\lambda)}$ such that $\mathsf{F}(x) = \mathsf{F}(x')$. This is a reasonable constraint because A has the option of outputting $x'=x$ if it does not find a true collision. We consider A's goal to be maximizing the entropy of its output $x'=A(x)$, given a random input x.

It follows directly that if A is *computationally unbounded*, then the optimum is exactly equal to the real entropy.

Proposition 3.16 (See [HHR+20])**.** *Let* $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$*. Then the real Shannon entropy of* F^{-1} *equals the maximum of* $\mathrm{H}(\mathsf{A}(X;R) \mid X)$ *over all (computationally unbounded)* F*-collision finders* A*, where the random variable* X *is uniformly distributed in* $\{0,1\}^{n(\lambda)}$ *and* R *is uniformly random coin tosses for* A*. That is,*

$$\mathrm{H}(X \mid \mathsf{F}(X)) = \max_{\mathsf{A}} \mathrm{H}(\mathsf{A}(X;R) \mid X),$$

where the maximum is taken over all F*-collision finders* A*.*

Proof. Entropy is maximal when the random variable is flat. Thus, the "optimal" F-collision finder A that maximizes $\mathrm{H}(\mathsf{A}(X) \mid X)$ is the algorithm $\widetilde{\mathsf{A}}$ that, on input x, outputs a uniformly random element of $\mathsf{F}^{-1}(\mathsf{F}(x))$. Then

$$\mathrm{H}(\widetilde{\mathsf{A}}(X;R) \mid X) = \mathbb{E}[\log |\mathsf{F}^{-1}(\mathsf{F}(X))|] = \mathrm{H}(X \mid \mathsf{F}(X)).$$

□

The notion of *accessible entropy* simply restricts the above to PPT algorithms. We consider both Shannon and max-entropy variants (since we aim to upper bound the accessible entropy, we do not consider the min-entropy variant).

Definition 3.17 (Accessible Entropy [HHR+20])**.** *Let $n(\lambda)$ be a security parameter and $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$ a function. We say that F^{-1} has* accessible Shannon entropy *at most k if for every* PPT F*-collision-finder A, we have*

$$\mathrm{H}(\mathsf{A}(X;R) \mid X) \leq k$$

for all sufficiently large $n(\lambda)$, where the random variable X is uniformly distributed on $\{0,1\}^{n(\lambda)}$ and R is uniformly random coin tosses for A.

*We say that F^{-1} has p-*accessible max-entropy *at most k if for every* PPT F*-collision-finder A, there exists a family of sets $\{\mathcal{L}(x)\}_{x\in\mathrm{Supp}(X)}$ each of size at most 2^k, such that*

$$\mathbb{P}[\mathsf{A}(X;R) \in \mathcal{L}(X)] \geq 1 - p$$

for all sufficiently large $n(\lambda)$, where the random variable X is uniformly distributed on $\{0,1\}^{n(\lambda)}$ and R is uniformly random coin tosses for A. In addition, if $p = \mathrm{neg}(n(\lambda))$ for some negligible function $\mathrm{neg}(\cdot)$, then we simply say that F^{-1} has accessible max-entropy *at most k.*

It is straightforward to verify that, ignoring negligible terms, the accessible Shannon entropy of F^{-1} is at most its accessible max-entropy, i.e., if the accessible max-entropy of F^{-1} is at most k, then its accessible Shannon entropy is at most k. We will later, in Section 3.4.3, introduce an in-between variant of accessible entropy that is larger than Shannon but smaller than max.

Having an upper bound on accessible entropy is useful as an intermediate step towards constructing UOWHFs since accessible max-entropy 0 is equivalent to target collision resistance (on random inputs).

Definition 3.18 (q-collision-resistant)**.** *Let $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$ be a function and $q = q(n) \in [0,1]$. We say that F is q-*collision-resistant on random inputs *if for every* PPT F*-collision-finder A,*

$$\mathbb{P}[\mathsf{A}(X;R) = X] \geq q,$$

for all sufficiently large $n(\lambda)$, where the random variable X is uniformly distributed on $\{0,1\}^{n(\lambda)}$ and R is uniformly random coin tosses for A. In addition, if $q = 1 - \mathrm{neg}(n(\lambda))$ for some negligible function $\mathrm{neg}(\cdot)$, we say that F is collision-resistant on random inputs.

Lemma 3.19 (See [HHR+20])**.** *Let $n(\lambda)$ be a security parameter and $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$ be a function. Then, for any $p = p(n) \in (0,1)$, the following statements are equivalent:*

(1) F^{-1} *has* p*-accessible max-entropy* 0*.*

(2) F *is* $(1-p)$*-collision-resistant on random inputs.*

In particular, F^{-1} *has accessible max-entropy* 0 *iff* F *is collision-resistant on random inputs.*

Proof. Note that (1) implies (2) follows readily from the definition. To see that (2) implies (1), simply take $\mathcal{L}(x) = \{x\}$. □

While bounding p-accessible max-entropy with negligible p is our ultimate goal, one of our constructions will work by first giving a bound on accessible Shannon entropy, and then deducing a bound on p-accessible max-entropy for a value of $p < 1$ using the following lemma.

Lemma 3.20 (See [HHR+20])**.** *Let* $n(\lambda)$ *be a security parameter and* $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$ *be a function. If* F^{-1} *has accessible Shannon entropy at most* k*, then* F^{-1} *has* p*-accessible max-entropy at most* $k/p + O(2^{-k/p})$ *for any* $p = p(n) \in (0,1)$*.*

Proof. Fix any PPT F-collision-finder A. From the bound on accessible Shannon entropy, we have that $\mathrm{H}(\mathsf{A}(X;R) \mid X) \leq k$. Applying Markov's inequality, we have

$$\mathbb{P}_{x \xleftarrow{R} X, r \xleftarrow{R} R}\left[\underset{\mathsf{A}(X;R)|X}{\mathrm{H}}(\mathsf{A}(x;r) \mid x) \leq k/p\right] \geq 1-p.$$

Take $\mathcal{L}(x)$ to be the set:

$$\mathcal{L}(x) = \{x\} \cup \left\{x' \colon \underset{\mathsf{A}(X;R)|X}{\mathrm{H}}(x' \mid x) \leq k/p\right\}.$$

We may rewrite $\mathcal{L}(x)$ as $\{x\} \cup \left\{x' \colon \mathbb{P}_r[\mathsf{A}(x;r) = x'] \geq 2^{-k/p}\right\}$. Then it follows $|\mathcal{L}(x)| \leq 2^{k/p} + 1$ and thus F^{-1} has p-accessible max-entropy at most $k/p + O(2^{-k/p})$. □

Once we have a bound on p-accessible max-entropy for some $p < 1$, we must apply several transformations to obtain a function with a good bound on $\mathrm{neg}(n)\lambda))$-accessible max-entropy.

3.4.3 Accessible Average Max-Entropy

The second construction in [HHR+20] (which achieves better parameters) starts with a bound on a different average-case form of accessible entropy, which is stronger than bounding the accessible Shannon entropy. The benefit of this notion is that it can be converted more efficiently to $\mathrm{neg}(n(\lambda))$-accessible max-entropy, by simply taking repetitions.

To motivate the definition, recall that a bound on accessible Shannon entropy means that the sample entropy $\mathrm{H}_{\mathsf{A}(X;R)|X}(x' \mid x)$ is small on average over $x \xleftarrow{R} X$ and $x' \xleftarrow{R} \mathsf{A}(x;R)$. This sample entropy may depend on both the input x and the output x' by the adversary (which in turn may depend on its coin tosses). A stronger requirement is to say that we have upper bounds $k(x)$ on the sample entropy that depend *only on* x. The following definition captures this idea, thinking of $k(x) = \log|\mathcal{L}(x)|$.

Definition 3.21 (Accessible Average Max-Entropy [HHR+20])**.** *Let $n(\lambda)$ be a security parameter and $\mathsf{F}\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m$ a function. We say that F^{-1} has* accessible average max-entropy *at most k if for every* PPT F*-collision-finder A, there exists a family of sets $\{\mathcal{L}(x)\}_{x\in\mathrm{Supp}(X)}$ and a negligible function $\mathrm{neg} = \mathrm{neg}(n(\lambda))$ such that $x \in \mathcal{L}(x)$ for all $x \in \mathrm{Supp}(X)$, $\mathbb{E}[\log|\mathcal{L}(X)|] \leq k$ and*

$$\mathbb{P}[\mathsf{A}(X;R) \in \mathcal{L}(X)] \geq 1 - \mathrm{neg}(n(\lambda)),$$

for all sufficiently large $n(\lambda)$, where the random variable X is uniformly distributed on $\{0,1\}^{n(\lambda)}$ and R is uniformly random coin tosses for A.

It is easy to verify that, ignoring negligible terms, the accessible average max-entropy of F^{-1} is at least its accessible Shannon entropy and at most its accessible max-entropy.

3.5 Hash Functions

Consider the family of functions $\mathsf{F} = \left\{f\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m\right\}$. Here, F is *explicit* if given the description of a function $f \in \mathsf{F}$ and $x \in \{0,1\}^{n(\lambda)}$, the value $f(x)$ can be computed in time $\mathrm{poly}(n,m)$. F is *constructible* if it is explicit and there is a PPT that given $x \in \{0,1\}^{n(\lambda)}$, and $y \in \{0,1\}^m$, outputs a random $f \xleftarrow{R} \mathsf{F}$ such that $f(x) = y$.

We will use two types of (combinatorial) hash functions.

3.5.1 Two-Universal Hashing

Definition 3.22 (Two-universal function family)**.** *A function family $\mathcal{H} = \{h\colon \mathcal{D} \mapsto \mathcal{R}\}$ is* two-universal *if for all $x \neq x' \in \mathcal{D}$, it holds that $\mathbb{P}_{h\xleftarrow{R}\mathcal{H}}[h(x) = h(x')] \leq 1/|\mathcal{R}|$.*

An example of such a function family is the set $\mathcal{H}_{s,t} = \{0,1\}^{s\times t}$ of Boolean matrices, where for $h \in \mathcal{H}_{s,t}$ and $x \in \{0,1\}^s$, we let $h(x) = h \times x$ (i.e., the matrix vector product over GF_2). Another canonical example is $\mathcal{H}_{s,t} = \{0,1\}^s$ defined by $h(x) := h \cdot x$ over $\mathrm{GF}(2^s)$, truncated to its first t bits.

A useful application of two-universal hash functions is to convert a source of high Rényi entropy to a (close to) uniform distribution.

Lemma 3.23 (Leftover hash lemma [ILL89, IZ89])**.** *Let X be a random variable over $\{0,1\}^{n(\lambda)}$ with $\mathrm{H}_2(X) \geq k$, let $\mathcal{H} = \left\{g\colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^m\right\}$ be two-universal, and let $H \xleftarrow{R} \mathcal{H}$. Then*

$$\mathrm{SD}((H, H(X)), (H, U_m)) \leq \frac{1}{2} \cdot 2^{(m-k)/2},$$

where U_m is uniform over $\{0,1\}^m$.

3.5.2 Many-Wise Independent Hashing

Definition 3.24 (ℓ-wise independent function family)**.** *A function family $\mathcal{H} = \{h\colon \mathcal{D} \mapsto \mathcal{R}\}$ is* ℓ-wise independent *if for any distinct $x_1, \ldots, x_\ell \in \mathcal{D}$, it holds that $(H(x_1), \ldots, H(x_\ell))$ for $H \xleftarrow{R} \mathcal{H}$ is uniform over $\mathcal{R}^\ell$.*

The canonical example of such an ℓ-wise independent function family is $\mathcal{H}_{s,t,\ell} = (\{0,1\}^s)^\ell$ defined by

$$(h_0, \ldots, h_{\ell-i})(x) := \sum_{0 \le i \le \ell-1} h_i \cdot x^i,$$

over $\mathrm{GF}(2^s)$, truncated to its first t bits.

Notice that, for $\ell > 1$, an ℓ-wise independent function family is also two-universal, but ℓ-wise independent function families, in particular with larger value of ℓ, have stronger guarantees on their output distribution compared with two-universal hashing.

3.6 One-Way Functions

A random oracle is a random function drawn uniformly randomly from the set of all possible functions (over specific input and output domains). Random oracle models are commonly used in cryptographic construction [BR93] to obtain guarantees of collision resistance. However, we can rely on one-way functions to provide such guarantees.

Let us recall the standard definition of one-way functions.

Definition 3.25 (one-way functions)**.** *A polynomial-time computable* $f\colon \{0,1\}^\lambda \mapsto \{0,1\}^*$ *is* one-way *if for every* PPT A

$$\mathbb{P}_{y \xleftarrow{R} f(U_{s(\lambda)})}\Big[\mathsf{A}(1^\lambda, y) \in f^{-1}(y)\Big] = \mathrm{neg}(\lambda). \tag{1}$$

Without loss of generality, it can be assumed that $s(\lambda) = \lambda$ and f is length-preserving (i.e., $|f(x)| = |x|$).

However, we actually only need a weaker notion of one-way functions, namely *distributional one-way functions*. Such a function is easy to compute, but it is hard to compute uniformly random preimages of random images.

Definition 3.26. *A polynomial-time computable* $f\colon \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ *is* distributional one-way*, if there exists a* $p \in \mathrm{poly}$ *such that*

$$\mathrm{SD}\Big((x, f(x))_{x \xleftarrow{R} \{0,1\}^{n(\lambda)}}, (\mathsf{A}(1^\lambda, f(x)), f(x))_{x \xleftarrow{R} \{0,1\}^{n(\lambda)}}\Big) \ge \frac{1}{p(\lambda)}$$

for any PPTM A *and large enough* $n(\lambda)$*.*

3.7 Collision Resistance

To obtain guarantees of unforgeability, we can rely on collision resistance of hash functions [KL14, HPS08]. However, *universal one-way hash functions*, as introduced by [NY89], are a weaker form of *collision-resistant hash functions*.

Definition 3.27 (Collision-Resistant Functions)**.** *A function family* $\mathcal{F}$ *is* collision-resistant *if given a randomly chosen function* $f \in \mathcal{F}$*, it is infeasible to find any pair of distinct inputs* x, x' *such that* $f(x) = f(x')$*.*

Universal one-way hash functions only require *target collision resistance*, where the adversary must specify one of the inputs x before seeing the description of the function f.

Definition 3.28 (Universal one-way hash functions)**.** *Let* $(x, \mathsf{state}) \xleftarrow{R} \mathsf{A}(1^\lambda)$.

A family of functions $\mathcal{F}_\lambda = \left\{\mathsf{F}_z \colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^{m(\lambda)}\right\}_{z \in \{0,1\}^\lambda}$*, for* $n(\lambda), m(\lambda) \in \mathrm{poly}(\lambda)$*, is a family of* universal one-way hash functions *if it satisfies:*

Efficiency: *given* $z \in \{0,1\}^\lambda$ *and* $x \in \{0,1\}^{n(\lambda)}$*,* $\mathsf{F}_z(x)$ *can be evaluated in time* $\mathrm{poly}(\lambda)$*.*

Shrinking: $m(\lambda) < n(\lambda)$*.*

Target Collision Resistance: *the probability that a* PPT *adversary* A *succeeds in the following game is negligible in* λ*:*

i. *Let* $(x, \mathsf{state}) \xleftarrow{R} \mathsf{A}(1^\lambda)$.
- *Abort if* $(x, \mathsf{state}) \notin \{0,1\}^{n(\lambda)} \times \{0,1\}^*$.

ii. *Let* $z \xleftarrow{R} \{0,1\}^\lambda$.

iii. *Let* $x' \xleftarrow{R} \mathsf{A}(\mathsf{state}, z)$.
- *Abort if* $x' \notin \{0,1\}^{n(\lambda)}$.

iv. A succeeds *if* $x \neq x'$ *and* $\mathsf{F}_z(x) = \mathsf{F}_z(x')$.

For universal one-way hash function family, $\mathcal{F}_\lambda = \left\{\mathsf{F}_z \colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^{m(\lambda)}\right\}$, a function $\mathsf{F}_z \colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^{m(\lambda)}$ has *input length* $n(\lambda)$, *key length* λ, and *output length* $m(\lambda)$.

The $\leftarrow$ notation in the definition above corresponds to the following: on security parameter 1^λ, algorithm A first samples an element x in the function family input domain. Then given the description of a function F_z uniformly drawn from the family, algorithm A has to find a collision with x: an element $x' \neq x$ that F_z maps to the same output value.

3.8 Coding Theory Basics

We review basic notions in coding theory, most of which can be found in the text of MacWilliams and Sloane [MS78].

The primary purpose of codes is to correct errors that occur over noisy communication channels. Messages are encoded from a block of k message symbols $\mathbf{u} = u_1 u_2 \ldots u_k$ into a *codeword* $\mathbf{x} = x_1 x_2 \ldots x_n$ where $n \geq k$. A codeword can be decoded into the original message even if some of the message symbols are corrupted. We measure the difference between messages or codewords in terms of Hamming distance.

Definition 3.29 (Hamming Distance)**.** *For alphabet* Σ *(e.g.,* $\Sigma = \{0,1\}$*) and* $\mathbf{x}, \mathbf{y} \in \Sigma^n$*, define the* Hamming distance *between* $\mathbf{x}$ *and* $\mathbf{y}$ *as:*

$$\mathsf{dist}(\mathbf{x}, \mathbf{y}) := |\{i \in [n] : x_i \neq y_i\}|.$$

Now we define an error-correcting code.

Definition 3.30 (Error-Correcting Code)**.** *For an alphabet* Σ*, an* $[n, k, d]_\Sigma$ error-correcting code *is a 2-tuple* $(\mathsf{Encode}, \mathsf{Decode})$ *algorithm where* $\mathsf{Encode} : \Sigma^k \to \Sigma^n$ *is an encoding algorithm such that for all* $\mathbf{m}, \mathbf{m}' \in \Sigma^k$ *where* $\mathbf{m} \neq \mathbf{m}'$*,*

$$\mathsf{dist}(\mathsf{Encode}(\mathbf{m}), \mathsf{Encode}(\mathbf{m}')) \geq d$$

and $\mathsf{Decode} : \Sigma^n \to \Sigma^k$ *is the decoding algorithm such that, for all messages* $\mathbf{m} \in \Sigma^k$ *and* possibly erroneous *codewords* $\mathbf{c} \in \Sigma^n$*, we have:*

$$\mathsf{dist}(\mathsf{Encode}(m), \mathbf{c}) \leq e_{\max} \implies \mathsf{Decode}(\mathbf{c}) = \mathbf{m}$$

where $e_{\max} \leq (d-1)/2$ *is the maximum number of erroneous symbols that* Decode *can correct. We denote the codeword size* $|\mathbf{c}|$ *by* λ_c *(i.e., the security parameter for the codeword).*

We might write $[n, k, d]$ to denote the code $[n, k, d]_{\{0,1\}}$ over the binary alphabet.

For an error-correcting code $[n, k, d]_\Sigma$, n is the length of the code, k is the dimension, and d is the minimum distance.

Definition 3.31 (Minimum Distance)**.** *The* minimum distance *(or simply* distance*) of a code is the minimum Hamming distance between its codewords:*

$$d = \min_{\mathbf{u},\mathbf{v} \in \Sigma^k : \mathbf{u} \neq \mathbf{v}} \mathsf{dist}(\mathsf{Encode}(\mathbf{u}), \mathsf{Encode}(\mathbf{v})).$$

The number of errors that a code can correct is related to its distance.

Theorem 3.32 (Theorem 2 in [MS78])**.** *A code with minimum distance* d *can correct* $\lfloor \frac{1}{2}(d-1) \rfloor$ *errors. If* d *is even, the code can simultaneously correct* $\frac{1}{2}(d-1)$ *errors and detect* $d/2$ *errors.*

An important property of codes is the rate.

Definition 3.33 (Rate)**.** *For an error-correcting code* $[n, k, d]_\Sigma$*, the* rate *(or* efficiency*) is* $R = k/n$*.*

We often measure the efficacy of codes in terms of the distance and the rate. The core problem in coding theory is to find codes with large rate and large distance.

The following theorems give upper and lower bounds on the size of a code with a given minimum distance.

Theorem 3.34 (Singleton Bound; see Theorem 11 in [MS78])**.** *For error-correcting code with parameters* $[n, k, d]$*,* $n - k \geq d - 1$*.*

Theorem 3.35 (Gilbert-Varshamov Bound; see Theorem 12 in [MS78])**.** *There exists a binary linear code of length* n*, with at most* r *parity checks and minimum distance at least* d*, provided that*

$$1 + \binom{n-1}{1} + \cdots + \binom{n-1}{d-2} < 2^r.$$

These theorems imply the existence of good codes.

4 Security Definitions

For watermarking of language model outputs, we provide security definitions for publicly-detectable watermarking schemes. The definitions are derived from the main definitions of distortion-freeness and unforgeability given in previous works [FGJ+25, CGZ23, CG24]. In the next section, we show that the constructions satisfy the definitions.

4.1 Entities in Security Model

In most of the literature on watermarking of generative models, there are two distinct entities: the *model provider* and the *user*. The model provider trains the generative model and provides an interface to the model (e.g., through an API or direct user interface). The *user*, on the other hand, is responsible for sending prompts to the model provider. Often, the user can be modeled as an adversary against which the security and robustness properties must hold [Aar23, HKAAL23, Aar22].

In our work, we generalize this entity modeling approach. We make no distinction between a user and another language model to allow for interaction between artificial intelligence (AI) generative agents. There are two roles any generative agent can take on: *sender* or *receiver*. This allows us to model the interaction between the sender and receiver and import analyses from the literature on multi-round cryptography and for our results to apply to multi-round cryptography [Geh98, HHRS15, RRR19, AH91, GK96, Dam93].

4.1.1 Sender

The sender is a generative agent that is responsible for providing the response to a prompt sent by a receiver. This agent provides the following service: given a prompt, it returns the model output for that prompt based on its (already-trained) model. When necessary, an honest sender will run the watermarking protocol at generation time to product the model output. At setup time, the sender generates a secret key and public key. The secret key is used to generate the model output if the output should be watermarked. The public key is used to detect the presence of a watermark if the output is watermarked. The sender has (white-box) access to the weights of the model, as well as to the private information for the watermarking scheme (e.g., the secret keys). However, in this work, the sender does not need to have access to the weights of the model as modeled in some previous works [ZWL24, ZGC+24, ZALW24, DMFZ22, TCH23].

4.1.2 Receiver

The receiver is a generative agent: a human user or another language model. The receiver generates one or more prompts at different stages of the interaction. Each prompt consists of one or more tokens that the language model can interpret and recognize. i.e., tokens are part of the vocabulary of the language model. The receiver sends a prompt to the language model provider in exchange for the model output. The receiver receives the model output and can test for the presence of a watermark by running the public detection algorithm using the public key of the sender. A receiver can become a sender at a later stage during the interaction between the agents. Also, modeling the receiver interactions with the sender allows the results to apply to the information hiding literature [Mou03, Mou05, MK05, HLW+24, HLW+25].

4.2 Generative Model

Our results could apply for more general classes of generative models. However, we primarily assume that the generative model is an auto-regressive model:

Definition 4.1 (Auto-regressive Model)**.** *An auto-regressive model* Model *over token vocabulary* $\mathcal{T}$ *is a deterministic algorithm that takes in a prompt* $\boldsymbol{\rho} \in \mathcal{T}^*$ *and tokens previously output by the model* $\boldsymbol{t} \in \mathcal{T}^*$ *and outputs a probability distribution* $p = \mathsf{Model}(\boldsymbol{\rho}, \boldsymbol{t})$ *over* $\mathcal{T}$*.*

$\mathsf{GenModel}$ uses Model as an oracle to implement a generative model as shown in Algorithm 2. We use Model and $\mathsf{GenModel}$ in subsequent definitions and proofs. We use subscript notation as shorthand for the N input, i.e., $\mathsf{GenModel}_N(\boldsymbol{\rho}) = \mathsf{GenModel}(N, \boldsymbol{\rho})$.

Algorithm 2 $\mathsf{GenModel}$

input: N, $\boldsymbol{\rho}$
$\boldsymbol{t} \leftarrow ()$
for $i = 1$ **to** N **do**
  $\boldsymbol{t} \leftarrow \boldsymbol{t} \parallel \mathsf{LMDecode}(\mathsf{Model}(\boldsymbol{\rho}, \boldsymbol{t}))$
end for
output: $\boldsymbol{t}$

$\mathsf{LMDecode}$ is the specific decoding method that implements multinomial sampling algorithm. It is used to sample from Model. The $\mathsf{GenModel}$ procedure is used to iteratively generate any number of tokens. $\mathsf{LMDecode}$ is the specific decoding method.

4.3 Definitions

Now, we can formally define the publicly-detectable watermarking scheme, which should satisfy soundness, completeness, (weak) robustness, and the distortion-free property. The watermarking schemes in this paper meets these definitions.

4.3.1 Publicly-Detectable Watermarking Scheme

Definition 4.2 (Publicly-Detectable Watermarking Scheme)**.** *A* $(\Delta_s, \Delta_c, \Delta_r)$*-publicly-detectable watermarking scheme* $\mathsf{WatScheme}$ *for a model* Model *over token vocabulary* $\mathcal{T}$ *is a tuple of algorithms* $\mathsf{WatScheme} = (\mathsf{Setup}, \mathsf{Watermark}, \mathsf{Detect})$ *where:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$ *outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$ *with respect to the security parameter* λ*.* sk *is the secret key whereas* pk *is the public key.*
- $\mathsf{Watermark}_{\mathsf{sk}}(\boldsymbol{\rho}) \xrightarrow{R} \boldsymbol{t}$ *produces response output* $\boldsymbol{t} \in \mathcal{T}^*$ *given a prompt* $\boldsymbol{\rho} \in \mathcal{T}^*$*. The watermarking algorithm requires the secret key* sk*.*
- $\mathsf{Detect}_{\mathsf{pk}}(\boldsymbol{t}^*) \to \{\mathtt{true}, \mathtt{false}\}$ *outputs* `true` *or* `false` *given a candidate watermarked output* $\boldsymbol{t}^*$*. The detection algorithm requires the public key* pk*.*

The algorithms $(\mathsf{Setup}, \mathsf{Watermark}, \mathsf{Detect})$ can be implemented in time that is polynomial in the security parameter(s). A $\mathsf{WatScheme}$ scheme is considered secure if the following security definitions are met.

4.3.2 Soundness

Definition 4.3 (Soundness/Unforgeability)**.** *A* $\mathsf{WatScheme}$ *is* Δ_s*-sound if any adversary* D *cannot generate watermarked output given the public detection key and any polynomial number of genuinely-watermarked outputs. That is, the following must be satisfied:*

$$\mathbb{P}\left[\begin{array}{c} \mathsf{Detect}_{\mathsf{pk}}(\boldsymbol{t}^*) = \texttt{true} \wedge \\ \mathsf{non\_overlapping}_k(\boldsymbol{t}^*, \boldsymbol{t}_1, \boldsymbol{t}_2, \ldots) = \texttt{true} \end{array} : \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \boldsymbol{t}^* \leftarrow \mathsf{D}^{\mathsf{Watermark}_{\mathsf{sk}}(\cdot)}(\mathsf{pk}) \end{array}\right] \leq \mathrm{neg}(\lambda).$$

Here, the adversary is allowed to make a polynomial number of queries to the oracle $\mathsf{Watermark}_{\mathsf{sk}}(\cdot)$*. We use* $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots$ *to denote the watermarked output that the adversary receives as output when it queries the model* $\mathsf{Watermark}_{\mathsf{sk}}(\cdot)$*. The predicate* $\mathsf{non\_overlapping}_{\Delta_s}(\boldsymbol{t}^*, \boldsymbol{t}_1, \boldsymbol{t}_2, \ldots)$ *outputs* `true` *if* $\boldsymbol{t}^*$ *does not share a* Δ_s*-length window of tokens with any of the genuinely-watermarked outputs* $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots$ *and outputs* `false` *otherwise.*

Unforgeability as soundness We provided some intuition for the soundness definition. If the adversary manages to output $\boldsymbol{t}^*$ that is labeled as watermarked, it must be the case that the adversary copied a sufficiently long sequence of tokens from the genuinely-watermarked outputs it received from the model (i.e., $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots$). This implies that any attempted forgery of a watermarked message must contain an overwhelming portion of tokens from genuinely watermarked output. This notion of unforgeability is parametrized by the overlapping length Δ_s. The larger Δ_s is, the more sound our scheme is. Looking ahead, the main constructions are flexible in that, for any desired overlapping parameter Δ_s, the construction can be adapted to meet the corresponding soundness guarantee.

4.3.3 Completeness

Definition 4.4 (Completeness)**.** *A* $\mathsf{WatScheme}$ *is* Δ_c*-complete if for every prompt* $\boldsymbol{\rho}$ *and token sequence* $\boldsymbol{t} \in \mathcal{T}^*$ *of length* $|\boldsymbol{t}| \geq \Delta_c$*, it holds that*

$$\mathbb{P}\left[\mathsf{Detect}_{\mathsf{pk}}(\boldsymbol{t}) = \texttt{false} \quad : \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \boldsymbol{t} \leftarrow \mathsf{Watermark}_{\mathsf{sk}}(\boldsymbol{\rho}) \end{array}\right] \leq \mathrm{neg}(\lambda).$$

The Δ_c-completeness ensures that output of sufficient length that was watermarked with the honest protocol results in non-detection only with negligible probability. As noted by [FGJ+25], this definition is an asymmetric-key analogue of the symmetric-key completeness definition in [CGZ23].

As noted by previous work of [KGW+23, CGZ23], completeness is not straightforward to satisfy when the response to the prompt has low-entropy. However, the watermarking algorithm can acquire more "empirical entropy" via increasing the output length [CGZ23] or use error-correction to handle low-entropy periods [FGJ+25, CG24] during generation.

4.3.4 Robustnesss

Definition 4.5 (Robustness)**.** *A publicly-detectable watermarking scheme is Δ_r-robust if, for every prompt $\boldsymbol{\rho}$ and security parameter λ,*

$$\mathbb{P}\left[\mathsf{Detect}_{\mathsf{pk}}(\mathsf{D}(\boldsymbol{t})) = \texttt{false} \quad : \quad \begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \boldsymbol{t} \leftarrow \mathsf{Watermark}_{\mathsf{sk}}(\boldsymbol{\rho}) \end{array} \right] \leq \mathrm{neg}(\lambda)$$

where the adversary is allowed to transform the input tokens $\boldsymbol{t}$ however it pleases so long as a Δ_r-length contiguous sequence of tokens remains. Let $\boldsymbol{t}^*$ be the adversarially-modified output (i.e. $\boldsymbol{t}^* \leftarrow \mathsf{D}(\boldsymbol{t})$). Then, there must exist a Δ_r-length window of tokens in $\boldsymbol{t}^*$ that exactly matches a Δ_r-length window in $\boldsymbol{t}$.

The robustness definition is meant to ensure the following: as long as a Δ_r-length contiguous sequence of tokens is preserved, the watermarked is also preserved.

Relationships between Δ_s, Δ_c, and Δ_r As noted by the previous work of [FGJ+25], it must be that $\Delta_s \leq \Delta_c \leq \Delta_r$. Intuitively, the watermarking scheme requires at least Δ_c tokens to embed a watermark. Thus, any block of Δ_r consecutive tokens is guaranteed to contain a block of Δ_c tokens that embeds the watermark. Additionally, any adversary that forges an accepting watermarked output must copy a segment of at least Δ_s tokens from the observed watermarked test.

4.3.5 Distortion-Free Property

The notion of distortion-freeness ensures that the watermarking algorithm does not noticeably change the quality of the model output. Without the public detection key, no PPT machine can distinguish between watermarked output (from $\mathsf{D}^{\mathsf{Model},\mathsf{Watermark}}$) and non-watermarked output (from $\mathsf{D}^{\mathsf{Model},\mathsf{GenModel}}$).

Definition 4.6 (Distortion-freeness)**.** *A* WatScheme *is (computationally) distortion-free if, for all* PPT *distinguishers* D*,*

$$\left|\mathbb{P}\left[\mathsf{D}^{\mathsf{Model},\mathsf{GenModel}}(1^\lambda) = 1\right] - \mathbb{P}_{(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{Setup}(1^\lambda)}\left[\mathsf{D}^{\mathsf{Model},\mathsf{Watermark}}(1^\lambda) = 1\right]\right| \leq \mathrm{neg}(\lambda),$$

where λ is the security parameter.

The distortion-free property is very similar to the "undetectability" notion from previous work of [CGZ23]. We will refer to the notion as distortion-free so as not to confuse the reader with the property of public-detectability of the watermark scheme.

5 Constructions

Our constructions are similar to those of previous work of Fairoze, Garg, Jha, Mahloujifar, Mahmoody, and Wang [FGJ+25], which we build upon. However there are important differences, as follows.

i. **Random Oracles**: They assume access to a random oracle. In our work, we instead make the minimal assumption of one-way functions (specifically, one-way permutations) and use it to construct distortion-free and unforgeable watermarking schemes.

ii. **Inaccessible Entropy**: We leverage the framework of inaccessible entropy [HRVW09] to analyze the model output generation and detection phases. During the generation and detection phases, hash error-correcting functions (Definition 5.1) are used to encode the generated message-signature pair and recover the signature. The generated codewords would be pseudorandom to a computationally bounded adversary. The analysis of the construction that satisfies Definition 5.1 is done via the framework of inaccessible entropy. We believe that our techniques extend beyond this particular use-case for watermarking.

iii. **Min-Entropy Assumptions**: The previous work of [FGJ+25] assume min-entropy lower bounds on the token sampling distribution. We find that such min-entropy lower bound assumptions are not needed. In particular, our security guarantees do not assume that the min-entropy of the token sampling distribution is high. (Intuitively, this is because of how we use the hashing strategies. In addition, you can always amplify the min-entropy of the output via, for example, the direct product operation on the hashing mechanism. See Lemma A.7.) Another approach to circumvent the min-entropy assumption is to use the "empirical entropy" approach of [CGZ23, CG24] used to collect more entropy from the token sampling distribution.

5.1 High-Level Discussion on Approach

The construction has two major components: one for the generation of the auto-regressive model tokens and the other for the detection of whether the tokens are generated from watermarked schemes or not. The generation algorithm is *private* (i.e., a private key is used to generate the output) while the detection phase is *public* (i.e., the public key is needed for detection).

The constructions adhere to Definition 4.2 for publicly-detectable watermarks. (See Section 4 for the suite of security definitions.) The constructions for output generation and detection form a publicly-detectable watermark. See Algorithm 3, Algorithm 4, and Algorithm 7 for the setup, output generation, and detection algorithms, respectively.

Prior to watermarking or detection, the Setup algorithm (Algorithm 3) is used to initialize the secret key $(\mathsf{sk}, \mathsf{r})$ and public key $(\mathsf{pk}, \mathsf{r})$ where sk and pk are generated by the native signature key generation algorithm and r is a uniformly random string to seed the hash functions. See the Hash-and-Sign paradigm review in the preliminaries (Section 3.2.1). Theorem 3.5 shows that using an already-secure signature scheme for fixed-length messages and a target collision-resistant hash function, there exists a secure scheme for arbitrary-length messages. This brings positive news since language model outputs can be of arbitrary length. Thus, our approach for security would be to prove that the insecurity of our model would contradict the insecurity of secure signature schemes. Thus, our model must be secure (according to Definition 3.4).

The main watermarking scheme (Algorithm 4) uses a fixed-length prompt (denoted $\boldsymbol{\rho}$) to generate N total tokens. $\mathsf{GenModel}$ (Algorithm 2) is able to generate any number of (non-watermarked) tokens using the auto-regressive model Model. For some settings of $\ell, \lambda_{\mathbf{c}}$, exactly $(\ell + \ell \cdot \lambda_{\mathbf{c}})$ tokens are used to map the message (the N total tokens to be generated) to a signature. In other words, the $(\ell + \ell \cdot \lambda_{\mathbf{c}})$ tokens represent a message-signature pair. We only need one such for detectability of the watermark. Once the message-signature pair is planted in the sequence of tokens $\boldsymbol{t}$, then $N - |\boldsymbol{t}|$ more tokens can be generated to complete the response of length N. Note that N is an arbitrary length but the settings of $\ell, \lambda_{\mathbf{c}}$ have to be such that $N \geq (\ell + \ell \cdot \lambda_{\mathbf{c}})$. The procedure to generate the message-signature pair is imported from [FGJ+25] with a few modifications in how the codewords are generated. The use of error-correcting codes enables low-entropy periods. Further, the codes

Algorithm 3 Setup

1: **input:** 1^λ
2: $\mathsf{r} \xleftarrow{R} \{0,1\}^\lambda$
3: $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda)$
4: **output:** $(\mathsf{sk}, \mathsf{r}), (\mathsf{pk}, \mathsf{r})$

Setup denotes the key generation algorithm for the watermarking procedure. It relies on Gen, a key generation procedure for a digital signature scheme (see Section 3.2). Gen produces a secret key sk and public key pk. Also, a random string r is generated in Setup. r is used to seed the hash functions that will be used to implement the Hash-and-Sign paradigm (see Section 3.2.1) that will ensure the distortion-free property.

satisfy security properties to enforce the distortion-free property. See Definition 5.1.

The detection phase is accomplished via Algorithm 7. Essentially, a linear search (with a window determined by ℓ) is done over the N tokens until a message and signature pair can be recovered.

Next, we describe the assumptions we make.

5.2 Assumptions

Previous work of Fairoze et al. [FGJ+25] assumed that any contiguous block of ℓ tokens is lower bounded by a min-entropy threshold. That is, there exists some $t > 0$ such that no particular sample is more than 2^{-t} likely to happen in any ℓ-length contiguous group of tokens. There are few ways to alleviate this assumption. One immediate way is to increase the length ℓ until the "empirical entropy" is guaranteed to be at least (a constant multiple of) the security parameters. Then through an analysis similar to that used by Christ et al. [CGZ23], one can argue that the security can be broken with exponentially small probability in the security parameters. This is essentially what we accomplish: we can argue that the input to the hash function has enough min-entropy and the construction of the hash function allows for amplifying the min-entropy of the hash outputs (up to a point that does not violate the data processing inequality).

In the algorithms, the parameter ℓ effectively serves as a parameter to tune the trade-off between robustness and distortion-freeness. Higher ℓ values lead to more distortion-free text at the cost of robustness and vice versa. This has been discussed in [FGJ+25].

In order to avoid relying on random oracles, we use one-way permutations to construct hash error-correcting functions (Definition 5.1). First, the functions are used to enforce the distortion-free property of the watermarked (i.e., a computationally bounded adversary cannot tell the difference between watermarked and non-watermarked model outputs). These functions also enable tolerating low-entropy periods. In fact, the maximum number of low-entropy periods the scheme can tolerate is exactly the maximum number of errors that the underlying hash error-correcting functions scheme can correct (Lemma 5.3). Furthermore, there always exists constructions of Definition 5.1 that can withstand the maximum number of low-entropy periods (Lemma 5.4).

Algorithm 4 Watermark

input: $(\mathsf{sk}, \mathsf{r})$, $\boldsymbol{\rho}$, N, ℓ, $\lambda_{\mathbf{c}}$, $a_{\max}$, $e_{\max}$
$\boldsymbol{t} \leftarrow ()$
while $|\boldsymbol{t}| + (\ell + \ell \cdot \lambda_{\mathbf{c}}) < N$ **do**
  $\boldsymbol{t} \leftarrow \mathsf{GenerateMessageSignaturePair}(\boldsymbol{\rho}, \boldsymbol{t}, a_{\max}, e_{\max})$
end while
if $|\boldsymbol{t}| < N$ **then**
  $\boldsymbol{t} \leftarrow \boldsymbol{t} \parallel \mathsf{GenModel}_{N-|\boldsymbol{t}|}(\boldsymbol{\rho}, \boldsymbol{t})$
end if
output: $\boldsymbol{t}$

Watermark is the overarching watermarking algorithm. Given input prompt $\boldsymbol{\rho}$, the algorithm generates N watermarked tokens in response to the prompt.

Algorithm 5 GenerateMessageSignaturePair

input: $\boldsymbol{\rho}$, $\boldsymbol{t}$, $a_{\max}, e_{\max}$
$\boldsymbol{t} \leftarrow \boldsymbol{t} \parallel \mathsf{GenModel}_{\ell}(\boldsymbol{\rho})$
$\boldsymbol{\sigma} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{H}_{\mathsf{pk},1}(\mathsf{r} \parallel \boldsymbol{t}_{[-\ell:]}))$
$\mathbf{c} \leftarrow \mathsf{HashEncode}_{\mathsf{pk}}(\mathsf{r} \parallel \boldsymbol{t}_{[-\ell:]}, \boldsymbol{\sigma})$
$\mathbf{m}, \mathbf{c}_{\mathsf{prev}} \leftarrow (), ()$
$e \leftarrow 0$
while $\mathbf{c} \neq ()$ **do**
  $c, \mathbf{c} \leftarrow \mathbf{c}_{[0]}, \mathbf{c}_{[1:]}$
  $\boldsymbol{t}, \mathbf{m}, \mathbf{c}_{\mathsf{prev}} \leftarrow \mathsf{RejectSampleTokens}(c, \boldsymbol{t}, \mathbf{m}, \mathbf{c}_{\mathsf{prev}}, e, a_{\max}, e_{\max})$
end while
output: $\boldsymbol{t}$

GenerateMessageSignaturePair is the main procedure that is responsible for "planting" the message-signature pair gadget into the $\ell + \ell \cdot \lambda_{\mathbf{c}}$ tokens. The ℓ-length message is sampled directly from the underlying auto-regressive model. Then the message is signed to obtain $\boldsymbol{\sigma}$. Then an encoding of $\boldsymbol{\sigma}$ is obtained as $\mathbf{c}$ which will be embedded into $\ell \cdot \lambda_{\mathbf{c}}$ tokens via a rejection sampling procedure.

Algorithm 6 RejectSampleTokens

input: $c, \boldsymbol{t}, \mathbf{m}, \mathbf{c}_{\mathsf{prev}}, e, a_{\max}, e_{\max}$
$a \leftarrow 0$
$\mathbf{x}_{\text{best}}, d_{\text{best}} \leftarrow (), \infty$
repeat
  $\mathbf{x} \leftarrow \mathsf{GenModel}_{\ell}(\boldsymbol{\rho}, \boldsymbol{t})$
  $a \leftarrow a + 1$
  $d \leftarrow \mathsf{dist}(\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}}), c)$
  if $d < d_{\text{best}}$ **then**
    $d_{\text{best}}, \mathbf{x}_{\text{best}} \leftarrow d, \mathbf{x}$
  end if
  if $(a > a_{\max} \ \wedge \ e < e_{\max})$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}_{\text{best}}$
    $e \leftarrow e + 1$
    break
  end if
until $\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}}) = c$
$\mathbf{m} \leftarrow \mathbf{m} \parallel \mathbf{x}$
$\boldsymbol{t} \leftarrow \boldsymbol{t} \parallel \mathbf{x}$
$\mathbf{c}_{\mathsf{prev}} \leftarrow \mathbf{c}_{\mathsf{prev}} \parallel c$
output: $\boldsymbol{t}, \mathbf{m}, \mathbf{c}_{\mathsf{prev}}$

RejectSampleTokens implements the rejection sampling loop from the previous work of [FGJ+25]. It is used to generate ℓ tokens such that each contiguous block of ℓ tokens encodes one bit of information.

5.3 Output Generation

Algorithm 4 is used for generating the watermarked output. Note that the construction can be applied repeatedly to generate arbitrarily long outputs from the model. The core idea is to embed a message and a corresponding publicly-verifiable signature in the generated text. The message-signature pair should be extractable during detection. Once extracted, it can be verified using the public key.

Once a single message-signature pair is embedded in the output, the watermark has been planted in the output. To perform watermarking, the first step is to sample a fixed number of tokens determined by the parameter ℓ. Ideally, the entropy of the tokens will be large enough to plant the message-signature pair. But even if it is not large enough, we rely on error-correction to plant the message-signature pair regardless of the entropy requirement. Then the planted message-signature can be error-corrected during the detection phase. The ℓ tokens are generated using the prompt $\boldsymbol{\rho}$. The generated $\boldsymbol{t}$ is used to denote the ℓ generated from the underlying autoregressive model.

The generated $\boldsymbol{t}$ is then hashed, signed, and encoded. It is known that not every codeword is a pseudorandom string [CG24]. Therefore, directly embedding the codeword distorts the distribution of the resulting output. Then using implementations for Definition 5.1, we can generate codewords that satisfy the pseudorandomness property that prevents detectability of the model outputs. One direct way to handle the problem of regaining pseudorandomness is to use a one-time pad on the codeword. This is the approach taken by [FGJ+25]. In our work, we circumvent the use of one-time pads and random oracles.

To do so, we encode $\mathbf{c} \leftarrow \mathsf{HashEncode}_{\mathsf{pk}}(\mathsf{r} \parallel \boldsymbol{t}, \boldsymbol{\sigma})$ where $\mathsf{HashEncode}_{\mathsf{pk}}$ is the encoding function for the hash error-correcting primitive. The seed r is used for the hash functions to ensure computational indistinguishability. Then the pseudorandom signature codeword $\mathbf{c}$ is computed and embedded into the model output.

Then the construct attempts to embed each bit of the pseudorandom codeword into the block of tokens of length ℓ. If the length of each codeword is $\lambda_{\mathbf{c}}$, this results in a plant of length $\ell \cdot \lambda_{\mathbf{c}}$. The rejection sampling procedure of [FGJ+25] is used (Algorithm 6). A number of attempts (denoted by $a_{\max}$) are made to find the best next ℓ tokens that hash to the next 1 embedded bit. During such attempts, a fresh block of length ℓ is sampled. The tokens are rejected if the resulting hash does not match with the codeword bit: For the bit c that is part of the codeword, $\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}})$ is computed (for fresh token samples $\mathbf{x}$) until it matches with the bit c. After $a_{\max}$ attempts, a "noisy version" of c is used instead (i.e., $\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}})$). This is why the resulting codeword would need to be error-corrected eventually in the detection phase via Algorithm 7.

Even though the "planting" of bits of the codeword is done bit-by-bit, as noted by [FGJ+25], a contiguous sequence of bits of the codewords could be used instead. For simplicity and clarity, we focus on planting the codewords bit-by-bit. If more than one bit of the codeword is planted at a time, the procedure (Algorithm 6) is written to take advantage of the distance between $\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}})$ and the bits from the codeword.

The input to the hash $\mathsf{H}_{\mathsf{pk},2}$ depends on all previous inputs to hashes for the current signature codeword. Once the codeword bit matches with the output of the hash function (or after $a_{\max}$ attempts if the output does not match with the codeword bit), we accept the token block and move on to the next 1 bit of the signature codeword.

As done in [FGJ+25], at the end of the rejection sampling process, the signature will be embedded in $\ell \cdot \lambda_{\mathbf{c}}$ tokens where $\lambda_{\mathbf{c}}$ is the length of the signature codeword. This corresponds to embedded one message-signature pair in the generated output of the model. Depending on the preference of

Algorithm 7 Detect

1: **input:** $(\mathsf{pk}, \mathsf{r})$, N, ℓ, $\lambda_{\mathbf{c}}$, $\boldsymbol{t}'$
2: **for** $i \in \{0, \ldots, N - (\ell + \ell \cdot \lambda_{\mathbf{c}})\}$ **do**
3: $\quad \boldsymbol{t} \leftarrow \mathsf{H}_{\mathsf{pk},1}(\mathsf{r} \parallel \boldsymbol{t}'_{[i:i+\ell]})$
4: $\quad \mathbf{m}, \mathbf{c}' \leftarrow (), ()$
5: $\quad$ **for** $j \in \{0, \ldots, \lambda_{\mathbf{c}} - 1\}$ **do**
6: $\quad\quad \mathbf{m} \leftarrow \mathbf{m} \parallel \boldsymbol{t}'_{[(i+\ell+1)+(j\cdot\ell):(i+\ell+1)+((j+1)\cdot\ell)]}$
7: $\quad\quad \mathbf{c}' \leftarrow \mathbf{c}' \parallel \mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{c}')$
8: $\quad$ **end for**
9: $\quad \boldsymbol{\sigma} \leftarrow \mathsf{HashDecode}_{\mathsf{pk}}(\mathbf{c}', \mathsf{r} \parallel \boldsymbol{t}'_{[i:i+\ell]})$
10: $\quad$ **if** $\mathsf{Verify}_{\mathsf{pk}}(\boldsymbol{t}, \boldsymbol{\sigma}) = \texttt{true}$ **then**
11: $\quad\quad$ **output:** `true`
12: $\quad$ **end if**
13: **end for**
14: **output:** `false`

The watermark detection algorithm is Detect. Given a (potentially watermarked) input $\boldsymbol{t}'$, it searches for a single embedded message-signature pair that passes authentication. If one such pair is found, the input text is flagged as watermarked.

the watermark designer, the process can be repeated to embed multiple message-signature pairs for better security guarantees.

Also, $a_{\max}$ is the parameter that is used to control how many times fresh tokens can be resampled during the rejection sampling loop until the loop condition is satisfied. And $e_{\max}$ is the maximum number of errors that $(\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}})$ can correct. This quantity can be determined based on the construction of the code (i.e., based on the distance of the underlying error-correcting code).

5.4 Output Detection

As previously noted, even though the token generation process requires a secret key, the detection phase only requires the public key. Given some input that could be watermarked, Algorithm 7 performs an exhaustive search on ℓ-length tokens to extract a message-signature pair that is verified via the public verification scheme of the underlying signature $\mathsf{Verify}_{\mathsf{pk}}$.

Specifically, in Algorithm 7, we iterate over all token windows of length ℓ. Once the message $\boldsymbol{t}$ is assigned, the signature is iteratively reconstructed. Since we employed the encoding function from the hash error-correcting function family, we would need to apply the decoding function during the detection phase. Error correction is used to handle the cases where the entropy is low to embed the bits of the codewords. This is what Line 9, $\boldsymbol{\sigma} \leftarrow \mathsf{HashDecode}_{\mathsf{pk}}(\mathbf{c}', \mathsf{r} \parallel \boldsymbol{t}'_{[i:i+\ell]})$, accomplishes.

If the signature verification step passes at least once, then we know with overwhelming probability the text was watermarked (refer to Lemma 5.7). At the end of the procedure, if no message-signature pair passes verification, then we can conclude that the model was not watermarked (refer to Lemma 5.6). The security of the scheme depends on the underlying security of the signature scheme. See Section 3.2.

5.5 Encoding and Decoding

To obtain the security properties outlined in Section 4, we can rely on the hash-and-sign paradigm outlined in Section 3.2.1. Theorem 3.5 implies that as long we have access to an underlying signature scheme, we can get a secure scheme via $\boldsymbol{\sigma} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{H}_{\mathsf{pk},1}(\mathsf{r} \parallel \boldsymbol{t}_{[-\ell:]}))$ in Algorithm 5.

However, $\boldsymbol{\sigma}$ is a string of bits that cannot be directly embedded in the model output as that would render the model output detectable (and thus violate Definition 4.6). So, instead we can encode the signature $\boldsymbol{\sigma}$ into the model output so that the model remains distortion-free. However, we must also be able to recover the signature $\boldsymbol{\sigma}$ during the detection phase. This motivates our definition (Definition 5.1) and use of $\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}}$.

In the watermarking scheme, there are two methods for encoding and decoding into codewords that can preserve the distortion-free property. In Algorithm 5, after signing the message, the following line is used to encode the signature:

$$\mathbf{c} \leftarrow \mathsf{HashEncode}_{\mathsf{pk}}(\mathsf{r} \parallel \boldsymbol{t}_{[-\ell:]}, \boldsymbol{\sigma}).$$

In Algorithm 7, the following line is used to recover the signature during the detection phase:

$$\boldsymbol{\sigma} \leftarrow \mathsf{HashDecode}_{\mathsf{pk}}(\mathbf{c}', \mathsf{r} \parallel \boldsymbol{t}'_{[i:i+\ell]}).$$

Thus, $\mathsf{HashEncode}_{\mathsf{pk}}$ and $\mathsf{HashDecode}_{\mathsf{pk}}$ ensure that we can encode and decode the signature into codewords that satisfy target collision resistance.

Definition 5.1 ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$)**.** *Let λ be the security parameter and $z \in \{0,1\}^\lambda$. Also, let $\boldsymbol{\sigma}$ be a signature, $\mathbf{m}$ be a message, and $\mathbf{c}$ be a codeword. We denote* ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$) *as* Hash Error-Correcting *functions that satisfy the following:*

- ***Error-Correcting**: $\mathsf{HashDecode}_z(\mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma}), \mathbf{m}) = \boldsymbol{\sigma}$ with probability at least $1 - \mathrm{neg}(\lambda)$.*
- ***Target Collision Resistance**: For every codeword $\mathbf{c}$ and message-signature pair $(\mathbf{m}, \boldsymbol{\sigma})$ such that $\mathbf{c} \leftarrow \mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma})$, only with probability at most $\mathrm{neg}(\lambda)$ can a PPT adversary find $(\mathbf{m}', \boldsymbol{\sigma}')$ such that $(\mathbf{m}, \boldsymbol{\sigma}) \neq (\mathbf{m}', \boldsymbol{\sigma}')$ and $\mathbf{c} = \mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma}) = \mathsf{HashEncode}_z(\mathbf{m}', \boldsymbol{\sigma}')$.*

We show a construction of a family of ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$) functions, as follows.

Theorem 5.2 (See Theorem A.2)**.** *Let λ be the security parameter. Suppose there exists a one-way permutation $f : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$. Then, there exists a family of* ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$) *functions with codeword length $\widetilde{O}(\lambda^7)$. That is, for any $k \in \mathbb{N}$ such that $k \leq m(\lambda) < n(\lambda)$, there exists a family of functions*

$$\left\{\mathsf{HashEncode}_z : \{0,1\}^{n(\lambda)} \times \{0,1\}^k \mapsto \{0,1\}^{m(\lambda)}\right\}_{z \in \{0,1\}^\lambda},$$

$$\left\{\mathsf{HashDecode}_z : \{0,1\}^{m(\lambda)} \times \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^k\right\}_{z \in \{0,1\}^\lambda},$$

where $m(\lambda) = \widetilde{O}(\lambda^7)$ and satisfies the properties in Definition 5.1.

There are some main goals that Theorem 5.2 is meant to accomplish: (1) The message-signature pair should be able to be encoded and that, given the message, the signature should be able to be decoded; (2) The probability of finding message-signature pairs that result in a codeword collision should be negligible (with respect to the security parameter).

There are a few ways to construct $(\mathsf{HashEncode}_z, \mathsf{HashDecode}_z)$. But intuitively, it can be built "on top" of an error-correcting code $(\mathsf{Encode}, \mathsf{Decode})$ and hash function H_z where $\mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma}) = \mathsf{Encode}(\boldsymbol{\sigma}) \oplus \mathsf{H}_z(\mathbf{m})$ and $\mathsf{HashDecode}_z(\mathbf{c}, \mathbf{m}) = \mathsf{Decode}(\mathsf{H}_z(\mathbf{m}) \oplus \mathbf{c})$. Then clearly the collision-resistance of H_z would transfer to $(\mathsf{HashEncode}_z, \mathsf{HashDecode}_z)$. Furthermore, the error-correcting of $(\mathsf{Encode}, \mathsf{Decode})$ would transfer to the error-correcing abilities of $(\mathsf{HashEncode}_z, \mathsf{HashDecode}_z)$ since $\mathsf{H}_z(\mathbf{m}) \oplus \mathsf{H}_z(\mathbf{m})$ will equals the all-zeros string.

For some $\mathsf{pk} \in \{0,1\}^\lambda$, we note that the use of $(\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}})$ allows for the watermarking scheme to tolerate low-entropy prompts. A low-entropy period is a block of ℓ tokens that do not hash to the bit value of the codeword.

Lemma 5.3. *Let* $\mathsf{pk} \in \{0,1\}^\lambda$. *The number of low-entropy periods that the watermarking scheme can handle is at most the number of errors that* $(\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}})$ *can correct.*

Proof. Consider the rejection sampling loop in Algorithm 6. Fix r, $\mathbf{m}$, $\mathbf{x}$, $\mathbf{c}_{\mathsf{prev}}$. If $\mathbf{c}$ is the codeword, then an iteration corresponds to some $i \in [\lambda_{\mathbf{c}}]$ and $\mathbf{c}_i = c$. Recall that $\mathsf{H}_{\mathsf{pk},2}$ is target collision resistant.

In iteration i, at most $a_{\max}$ attempts are made to sample $\mathbf{x}$ such that the following condition is satisfied: $\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}}) = c$. After the (failed) attempts, instead of c, $1-c = \mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}})$ would be used to plant $\mathbf{x}$. This corresponds to an error occurring due to the low-entropy of the sampling distribution. In particular, instead of the encoding $\mathbf{c} = \mathbf{c}_1\mathbf{c}_2 \ldots \mathbf{c}_i \ldots \mathbf{c}_{\lambda_{\mathbf{c}}}$ being used to plant $\mathbf{x}$, $\mathbf{c}' = \mathbf{c}_1\mathbf{c}_2 \ldots (1-\mathbf{c}_i) \ldots \mathbf{c}_{\lambda_{\mathbf{c}}}$ is used instead.

An error has occurred on bit i of $\mathbf{c}$. And this occurred exactly because the output of the hash did not change after a few attempts (i.e., the sampling distribution from which $\mathbf{x}$ was sampled from has low-entropy). As a result, an error would occur as many times as there are low-entropy sampling periods. So, the maximum number of low-entropy periods the scheme can handle is at most the number of errors that $(\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}})$ can correct.

□

Lemma 5.4. *There exists* $(\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}})$ *function families such that the scheme* $\mathsf{WatScheme}$ *can withstand the maximum number of low-entropy periods.*

Proof. Fix a token sampling distribution from which $\mathsf{GenModel}$ generates tokens from in the $\mathsf{Watermark}$ procedure (Algorithm 4). Fix parameters $N, \ell \in \mathbb{N}$ in Algorithm 4. Then by Lemma 5.3, we just have to show the existence of codes that can correct as many errors as there are low-entropy periods. It might be helpful to consider $(\mathsf{HashEncode}_{\mathsf{pk}}, \mathsf{HashDecode}_{\mathsf{pk}})$ to be equivalent to:

$$\left\{\mathsf{HashEncode}_{z,p} : \{0,1\}^k \mapsto \{0,1\}^{m(\lambda)}\right\}_{z\in\{0,1\}^\lambda, p\in\{0,1\}^{n(\lambda)}} =$$
$$\left\{\mathsf{HashEncode}_z : \{0,1\}^{n(\lambda)} \times \{0,1\}^k \mapsto \{0,1\}^{m(\lambda)}\right\}_{z\in\{0,1\}^\lambda},$$

$$\left\{\mathsf{HashDecode}_{z,p} : \{0,1\}^{m(\lambda)} \mapsto \{0,1\}^k\right\}_{z\in\{0,1\}^\lambda, p\in\{0,1\}^{n(\lambda)}} =$$
$$\left\{\mathsf{HashDecode}_z : \{0,1\}^{m(\lambda)} \times \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^k\right\}_{z\in\{0,1\}^\lambda}.$$

Thus, $(\mathsf{HashEncode}_{z,p}, \mathsf{HashDecode}_{z,p})_{z\in\{0,1\}^\lambda, p\in\{0,1\}^{n(\lambda)}}$ are just error-correcting codes (Definition 3.30). By Theorem 3.32, codes with minimum distance d can correct at most $e = \lfloor \frac{1}{2}(d-1) \rfloor$ errors. For some $\ell \in \mathbb{N}$, note that the maximum number of low-entropy periods that $\mathsf{Watermark}$ (Algorithm 4) will encounter is the number of errors the error-correcting code can handle. So the code distance can be adjusted appropriately. By Theorem 3.35, for some $m(\lambda)$ there exists codes to instantiate $(\mathsf{HashEncode}_{z,p}, \mathsf{HashDecode}_{z,p})$ that can correct e errors which is equivalent to distance at least $2\lceil e \rceil + 1$. This can be accomplished by appropriately adjusting $m(\lambda) = \lambda_{\mathbf{c}}$ until it matches the bound given in Theorem 3.35 for the fixed distance of $2\lceil e \rceil + 1$.

□

5.6 Security and Robustness Properties

In this section, we show that the watermarking scheme $\mathsf{WatScheme} = (\mathsf{Setup}, \mathsf{Watermark}, \mathsf{Detect})$ defined via Algorithms 3, 4 and 7 satisfies security and robustness properties outlined in Section 4.

The analysis of robustness and security will follow from our use of hash functions in the algorithms: the hash error-correcting functions (Definition 5.1) and universal one-way hash functions (Definition 3.28).

Recall the hash functions are of the form:

$$\mathsf{H}_{\mathsf{pk},1} : \{0,1\}^* \mapsto \{0,1\}^{\lambda_{\boldsymbol{\sigma}}}, \quad \mathsf{H}_{\mathsf{pk},2} : \{0,1\}^* \mapsto \{0,1\},$$

where $\lambda_{\boldsymbol{\sigma}} = |\boldsymbol{\sigma}| \geq \lambda$, $\boldsymbol{\sigma}$ is the signature, and $\mathsf{H}_{\mathsf{pk},2}$ is the hash function that outputs a single bit (e.g., derived from any bit from the result of $\mathsf{H}_{\mathsf{pk},2} : \{0,1\}^* \mapsto \{0,1\}^{\lambda_{\mathbf{c}}}$), and

$$\mathsf{HashEncode}_{\mathsf{pk}} : \{0,1\}^* \times \{0,1\}^{\lambda_{\boldsymbol{\sigma}}} \mapsto \{0,1\}^{\lambda_{\mathbf{c}}},$$

where $\lambda_{\mathbf{c}} = |\mathbf{c}| \geq \lambda$, $\mathbf{c}$ is the codeword used to encode the message-signature pair.

Both $\mathsf{H}_{\mathsf{pk},1}, \mathsf{H}_{\mathsf{pk},2}$ will be instantiated as universal one-way hash functions which can be constructed via one-way permutations. Naor and Yung [NY89] showed that digital signatures can be constructed from universal one-way hash functions (UOWHF). In particular, they show that the random oracle assumption is not necessary to construct UOWHF. In a similar fashion, we show that the random oracle assumption is not necessary for publicly-verifiable watermarking. See Theorem B.2 that can be used to instantiate $\mathsf{H}_{\mathsf{pk},1}, \mathsf{H}_{\mathsf{pk},2}$.

$\mathsf{HashEncode}_{\mathsf{pk}}$ satisfies properties outlined in Definition 5.1. See Theorem 5.2 that can be used to instantiate $\mathsf{HashEncode}_{\mathsf{pk}}$. The analysis follows by building on the work of Haitner et al. [HHR+20, HHR+10] that provide the framework of inaccessible entropy.

We will prove the following theorem.

Theorem 5.5. *Suppose there exists a one-way function. Let $\lambda_{\mathbf{c}}$ be the length of the codewords in $\mathsf{WatScheme}$. For any $\lambda_{\mathbf{c}}, \ell, N \in \mathbb{N}$, the scheme $\mathsf{WatScheme}$ defined in Algorithms 3, 4 and 7 is an $(\ell, \ell + \ell \cdot \lambda_{\mathbf{c}}, 2(\ell + \ell \cdot \lambda_{\mathbf{c}}))$-publicly-detectable watermark that satisfies Definition 4.2.*

Theorem 5.5 follows immediately from the following:

- It is known that one-way functions can be used to construct the underlying digital signature that satisfies Definition 3.3 [Lam79, Rom90].
- Lemmas 5.6 to 5.9.

First, we consider soundness. Output that is not watermarked by WatScheme should not be marked as watermarked during the detection phase.

Lemma 5.6 (Soundness)**.** WatScheme *is an ℓ-sound (Definition 4.3) publicly-detectable watermarking scheme.*

Proof. We can show this via contradiction: the violation of the soundness property (Definition 4.3) would lead to a violation of the unforgeability property of the underlying signature scheme.

Recall that from Definition 3.3, for any message $m \in \{0,1\}^*$, $\mathsf{Sign}_{\mathsf{sk}}(m)$, $\mathsf{Verify}_{\mathsf{pk}}(m, \boldsymbol{\sigma})$ are for signing and verifying the message. Theorem 3.5 implies the unforgeability property (Definition 3.4) of the hash-and-sign paradigm outlined in Algorithm 1. Let D be an adversary that breaks the ℓ-soundness of the watermarking scheme. Then consider the following adversary D' that can break the unforgeability of the underlying signature scheme: D' will use D to break Definition 4.3. D' will also use an external signing oracle to obtain new signatures that it feeds to D to break the soundness guarantee in Definition 4.3 with a greater-than-negligible probability. This means that D can output contiguous blocks of tokens which a detector will say is watermarked. The detector will say the block of tokens is watermarked if and only if a message-signature pair (denote by $(\boldsymbol{t}, \boldsymbol{\sigma})$) has been extracted. By Definition 4.3, no ℓ-length sub-string of the output of D is a sub-string of the watermarked outputs that D has received previously. But recall that $\boldsymbol{t}$ is the output of the hash $\mathsf{H}_{\mathsf{pk},1}$ on ℓ consecutive tokens. By the target collision-resistance property of $\mathsf{H}_{\mathsf{pk},1}$, with probability at least $1 - \mathrm{neg}(\lambda)$, $\boldsymbol{t}$ has not previously appeared in any of the oracle queries of D'. This must mean that D' is able to violate Definition 3.4. Thus D' breaks the unforgeability of the underlying signature scheme.

□

Next, we consider the completeness guarantee. Output that is watermarked by WatScheme should be detectable by WatScheme.

Lemma 5.7 (Completeness)**.** *Let $\lambda_{\mathbf{c}}$ be the length of the codewords in* WatScheme*. For any $\lambda_{\mathbf{c}}, \ell, N \in \mathbb{N}$,* WatScheme *is a $(\ell + \ell \cdot \lambda_{\mathbf{c}})$-complete (Definition 4.4) publicly detectable watermarking scheme.*

Proof. To satisfy Definition 4.4, we need only show that with only with $\mathrm{neg}(\lambda)$ probability will watermarked output with $(\ell + \ell \cdot \lambda_{\mathbf{c}})$ tokens fail to be detected.

In Algorithm 5, $\mathbf{c} \leftarrow \mathsf{HashEncode}_{\mathsf{pk}}(\mathsf{r} \parallel \boldsymbol{t}_{[-\ell:]}, \boldsymbol{\sigma})$ is used to encode the last ℓ tokens and the signature $\boldsymbol{\sigma}$. Then an attempt is made to embed each bit of the codeword $\mathbf{c} = \mathbf{c}_1\mathbf{c}_2 \ldots \mathbf{c}_j \ldots \mathbf{c}_{\lambda_{\mathbf{c}}}$. Consider any $j \in [\lambda_{\mathbf{c}}]$ and fix $\mathsf{r}, \mathbf{m}, \mathbf{x}, \mathbf{c}_{\mathsf{prev}}$ in the jth iteration of going through the rejection sampling loop. Because of the (possibly low) entropy of the token sampling distribution from which $\mathbf{x}$ is drawn, during rejection sampling in Algorithm 6, the condition might not hold: $\mathsf{H}_{\mathsf{pk},2}(\mathsf{r} \parallel \mathbf{m} \parallel \mathbf{x} \parallel \mathbf{c}_{\mathsf{prev}}) = c = \mathbf{c}_j$. So, instead $\mathbf{c}' = \mathbf{c}'_1\mathbf{c}'_2 \ldots (1 - \mathbf{c}_j) \ldots \mathbf{c}'_{\lambda_{\mathbf{c}}}$ is used instead to plant $\mathbf{x}$. So the codeword $\mathbf{c}$ is transformed (by errors which correspond to low-entropy periods) to $\mathbf{c}'$. The codeword $\mathbf{c}$ will be recoverable from $\mathbf{c}'$ if the number of errors is within a constant of the distance of the code. And we know that by Lemma 5.3 and Lemma 5.4, during the detection phase, $\mathbf{c}$ can always be recovered from the noisy codeword $\mathbf{c}'$. The bad case is when the number of errors is greater than a constant of the distance (or length) of the code. However, this happens with probability at most $\mathrm{neg}(\lambda)$. This is because that case would correspond to producing $\mathbf{c}' = (1-\mathbf{c}_1)(1-\mathbf{c}_2)\ldots(1-\mathbf{c}_{i-1})(1-\mathbf{c}_i)\mathbf{c}_{i+1}\ldots\mathbf{c}_{\lambda_{\mathbf{c}}}$ for some $i \in [\lambda_{\mathbf{c}}]$ where i is within a constant multiple

of $\lambda_{\mathbf{c}}$ (e.g., $i = 1/2 \cdot \lambda_{\mathbf{c}}$ where i is the number of errors). However, by the target collision resistance of $\mathsf{H}_{\mathsf{pk},2}$ and $\mathsf{HashEncode}_{\mathsf{pk}}$, the chance of producing such $\mathbf{c}'$ will be $\mathrm{neg}(\lambda)$ which would guarantee that $\mathbf{c}$ can be recovered. Thus, Detect can, with probability at least $1 - \mathrm{neg}(\lambda)$, recover the original signature in the line $\boldsymbol{\sigma} \leftarrow \mathsf{HashDecode}_{\mathsf{pk}}(\mathbf{c}', \mathsf{r} \parallel \boldsymbol{t}'_{[i:i+\ell]})$ in Algorithm 7 and the verification stage will pass.

□

Here we discuss the (weak) robustness guarantee the scheme can tolerate.

Lemma 5.8 (Robustness)**.** $\mathsf{WatScheme}$ *is a* $2(\ell + \ell \cdot \lambda_{\mathbf{c}})$*-robust (Definition 4.5) publicly-detectable watermarking scheme.*

Proof. Definition 4.5 allows for an adversary to transform the text as long as a contiguous block of tokens remains that allows for extracting the message-signature pair. Once the message-signature pair is extracted, verification should pass (by the completeness guarantees provided by Lemma 5.7).

Let D be an adversary that has access to the watermarked output $\boldsymbol{t}$. Then if after the adversary's changes, $\mathsf{D}(\boldsymbol{t})$ contains at least $2(\ell + \ell \cdot \lambda_{\mathbf{c}})$ tokens from $\boldsymbol{t}$, then it is guaranteed that (with all but $\mathrm{neg}(\lambda)$ probability), a message-signature pair can be extracted from the output $\mathsf{D}(\boldsymbol{t})$ and the detection phase would still pass.

□

One of the most important properties for the watermarking scheme is distortion-freeness: a polynomial-time adversary should not be able to distinguish watermarked outputs from non-watermarked output. Definition 4.6 assumes that the adversary does not have access to the public key $(\mathsf{pk}, \mathsf{r})$. Clearly, distortion-freeness cannot hold when the adversary has access to $(\mathsf{pk}, \mathsf{r})$ (since it can run detect and confidently tell if the content is watermarked or not). See Definition 4.6 for the definition of the distortion-free property. We show that $\mathsf{WatScheme}$ achieves the property.

Lemma 5.9 (Distortion-freeness)**.** $\mathsf{WatScheme}$ *is a distortion-free publicly-detectable watermarking scheme (according to Definition 4.6).*

Proof. To satisfy Definition 4.6, we have to show that only with probability at most $\mathrm{neg}(\lambda)$ can a poly-time adversary D distinguish model outputs from $\mathsf{GenModel}$ (non-watermarked) versus model outputs from $\mathsf{Watermark}$.

It is easy to see that an adversary D can distinguish $\mathsf{GenModel}$ and $\mathsf{Watermark}$ if and only if it can distinguish their token sampling process.

Consider the following sampling processes:

(A) Sample tokens $\mathbf{x}$ through $\mathsf{GenModel}$.

(B) Sample tokens $\mathbf{x}'$ through $\mathsf{Watermark}$ (i.e., uses $\mathsf{Sign}_{\mathsf{sk}}$, $\mathsf{H}_{\mathsf{pk},1}$, $\mathsf{H}_{\mathsf{pk},2}$, $\mathsf{HashEncode}_{\mathsf{pk}}$).

Since $\mathbf{x}'$ could have been generated from $\mathsf{GenModel}$, the only way for the adversary to distinguish the sampling processes corresponding to items (A) and (B) is to simulate the process of generating $\mathbf{x}'$ or of detecting $\mathbf{x}'$. First, we discuss the adversary attempting to simulate generating $\mathbf{x}'$ and then discuss the adversary attempting to simulate the process of detecting $\mathbf{x}'$. In both cases, the chance of success would be negligible (in the security parameter).

In Algorithm 5, with access to $\boldsymbol{\sigma}$, without access to the secret key, the chance of obtaining a collision of one of $\mathsf{sk}, \mathsf{pk}, \mathsf{r}$ is negligible. So the chance of obtaining the codeword $\mathbf{c}$ is negligible. We

can interpret $\mathsf{H}_{\mathsf{pk},2}$ to be of the form $\mathsf{H}_{\mathsf{pk},2} : \{0,1\}^* \mapsto \{0,1\}^{\lambda_{\mathbf{c}}}$ since it is used to match tokens to each bit of the codeword. By the target collision properties of $\mathsf{H}_{\mathsf{pk},2}$, $\mathsf{HashEncode}_{\mathsf{pk}}$, the chance of obtaining the codeword, or using it to sample $\mathbf{x}'$, would be negligible.

Next, we focus on simulating the detection algorithm (Algorithm 7) but without access to the public key $(\mathsf{pk}, \mathsf{r})$. For the adversary let $\boldsymbol{t}' = \mathbf{x}'$ in Algorithm 7. By the target collision resistance of $\mathsf{H}_{\mathsf{pk},1}$, it is infeasible (except with negligible probability) for the adversary to generate $(\bar{\mathsf{r}}, \overline{\mathsf{pk}}) \neq (\mathsf{r}, \mathsf{pk})$ such that $\boldsymbol{t} \leftarrow \mathsf{H}_{\mathsf{pk},1}(\mathsf{r} \parallel \boldsymbol{t}')$, $\boldsymbol{t} \leftarrow \mathsf{H}_{\bar{\mathsf{pk}},1}(\bar{\mathsf{r}} \parallel \boldsymbol{t}')$. So, except with negligible probability (in the security parameter λ), the adversary cannot learn any of $\boldsymbol{t}, \mathsf{pk}, \mathsf{r}$ nor values that obtain hash collisions for $\boldsymbol{t}$. The adversary can attempt to reconstruct $\boldsymbol{\sigma}$ (via the line $\boldsymbol{\sigma} \leftarrow \mathsf{HashDecode}_{\mathsf{pk}}(\mathbf{c}', \mathsf{r} \parallel \boldsymbol{t}')$ in Algorithm 7). Again, by the target collision resistance of $\mathsf{HashDecode}_{\mathsf{pk}}$, only with negligible probability can the adversary generate $(\bar{\mathsf{r}}, \overline{\mathsf{pk}}) \neq (\mathsf{r}, \mathsf{pk})$ such that $\boldsymbol{\sigma} \leftarrow \mathsf{HashDecode}_{\overline{\mathsf{pk}}}(\mathbf{c}', \bar{\mathsf{r}} \parallel \boldsymbol{t}')$. (Even if the adversary has access to $\boldsymbol{\sigma}$, its goal is to get a hash collision to $\boldsymbol{\sigma}$ or to learn $(\mathsf{pk}, \mathsf{r})$.) The only "weak" link here is from $\mathbf{c}'$ which the adversary can attempt to reconstruct bit by bit. However, due to the target collision resistance of $\mathsf{H}_{\mathsf{pk},2}$, the adversary cannot have chance of greater than $\mathrm{neg}(\lambda)$ of correctly obtaining $\mathbf{c}'$ without knowledge of the public key. Thus, only with negligible probability will the detection phase pass on $\mathbf{x}'$.

So, without the private key $(\mathsf{sk}, \mathsf{r})$ or public key $(\mathsf{pk}, \mathsf{r})$, a PPT adversary cannot simulate the generation or detection phases for $\mathbf{x}'$. So, the sampling process for (A) and (B) will be computationally indistinguishable and the distortion-free property would be satisfied.

□

6 Conclusion

We constructed *distortion-free* and *unforgeable* watermarks for language models and generative agents. The distortion-free property guarantees that the sampling process of the watermarking scheme is computationally indistinguishable from the process of the non-watermarking scheme. That is, the adversary cannot detect any noticeable changes in the quality of the model output. Furthermore, no efficient adversary can forge the watermark. Our scheme embeds a (publicly-verifiable) digital signature into the language model output. The message and signature can be extracted during the detection phase. Any authorized entity with the public key can verify the message and signature. Our analysis relies on the minimal cryptographic assumption of one-way functions (specifically, permutations). This assumption implies a gap between the accessible entropy and real entropy of certain hash functions. These hash functions are used to instantiate the digital signature scheme for the language model. Furthermore, we have additional analysis on the use of error-correcting codes. The codes are also used to handle (possibly) low-entropy token sampling distributions.

There are several rooms for improvement in the scheme. For example, the detection phase could possibly be made much more efficient. *Also, is there an efficient way to embed more information beyond the message-signature pair (e.g., model version or date of creation) into the model output* (as in the information-theoretic information hiding literature)? We believe the use of inaccessible entropy for language model analysis could find applications beyond watermarking. In addition, the use of error-correcting codes in watermarking schemes is still under-explored. Constructing such codes that allow for robustness and indistinguishability remains an active area of study.

References

[AAC+24] Omar Alrabiah, Prabhanjan Ananth, Miranda Christ, Yevgeniy Dodis, and Sam Gunn. Ideal pseudorandom codes. Cryptology ePrint Archive, Paper 2024/1840, 2024. 4

[Aar22] Scott Aaronson. My AI Safety Lecture for UT Effective Altruism. https://scottaaronson.blog/?p=6823, November 2022. Accessed May 2024. 4, 20

[Aar23] Scott Aaronson. Neurocryptography. Invited Plenary Talk at Crypto'2023, 2023. 2, 4, 20

[AF21] Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 121–140. IEEE, 2021. 6

[AH91] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, 1991. Preliminary version in *FOCS'87*. 20

[Ber73] E. Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590–592, 1973. 7

[Ber16] Daria Beresneva. Computer-generated text detection using machine learning: A systematic review. In *Natural Language Processing and Information Systems: 21st International Conference on Applications of Natural Language to Information Systems*, 2016. 2

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993. 4

[BG89] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *Annual International Cryptology Conference (CRYPTO)*, pages 194–211, 1989. 8

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2), May 2012. 7

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. 4

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference*

on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III, volume 11478 of *Lecture Notes in Computer Science*, pages 619–635. Springer, 2019. 4

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993. 17

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In *Annual International Cryptology Conference (CRYPTO)*, pages 470–484, 1997. 52

[CBZ+23] Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. On the possibilities of AI-generated text detection. *arXiv preprint arXiv:2304.04736*, 2023. 2

[CG24] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. Cryptology ePrint Archive, Paper 2024/235, 2024. 4, 7, 20, 22, 24, 28

[CGZ23] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *IACR Cryptol. ePrint Arch.*, page 763, 2023. 2, 4, 5, 20, 22, 23, 24, 25

[CHV15] Aloni Cohen, Justin Holmgren, and Vinod Vaikuntanathan. Publicly verifiable software watermarking. Cryptology ePrint Archive, Paper 2015/373, 2015. 7

[CT99] Christian Collberg and Clark Thomborson. Software watermarking: models and dynamic embeddings. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '99, page 311–324, New York, NY, USA, 1999. Association for Computing Machinery. 7

[Dam93] Ivan B. Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions. pages 100–109, 1993. 20

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 8, 9

[DHT12] Yevgeniy Dodis, Iftach Haitner, and Aris Tentes. On the instantiability of hash-and-sign rsa signatures. In *Theory of Cryptography (TCC)*, pages 112–132, 2012. 8

[DMFZ22] Long Dai, Jiarong Mao, Xuefeng Fan, and Xiaoyi Zhou. Deephider: A multi-module and invisibility watermarking scheme for language model. *arXiv preprint arXiv:2208.04676*, 2022. 2, 20

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008. 56

[DSG+24] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634:818–823, October 2024. 2

[EGM89] Shimon Even, Oded Goldreich, and Silvio Micali. On-line off-line digital signatures. In *Conference on the Theory and Application of Cryptology*, pages 263–275, 1989. 8

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Annual International Cryptology Conference (CRYPTO)*, pages 10–18, 1984. 8

[EOS06] D. Engelbert, R. Overbeck, and A. Schmidt. A summary of McEliece-type cryptosystems and their security. Cryptology ePrint Archive, Paper 2006/162, 2006. 4, 7

[FGJ+23] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. Cryptology ePrint Archive, Paper 2023/1661, 2023. 3

[FGJ+25] Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. *IACR Communications in Cryptology*, 1(4), 2025. 4, 20, 22, 23, 24, 25, 27, 28

[Geh98] Christian Gehrmann. Multiround unconditionally secure authentication. *Designs, Codes and Cryptography*, 15(1):67–86, 1998. 20

[GG24] Surendra Ghentiyala and Venkatesan Guruswami. New constructions of pseudorandom codes. Cryptology ePrint Archive, Paper 2024/1425, 2024. 4

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. pages 123–139, 1999. 2, 10

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996. 20

[GM24] Noah Golowich and Ankur Moitra. Edit distance robust watermarks for language models. Cryptology ePrint Archive, Paper 2024/898, 2024. 4

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 8

[GPT23] GPTZero. GPTZero — The Trusted AI Detector for ChatGPT, GPT-4, & More. https://gptzero.me/, 2023. Accessed: 2023-10-05. 2

[GSR19] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. GLTR: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*, 2019. 2

[HHR+10] Iftach Haitner, Thomas Holenstein, Omer Reingold, Salil Vadhan, and Hoeteck Wee. Universal one-way hash functions via inaccessible entropy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 616–637, 2010. 5, 12, 32, 44, 54

[HHR+20] Iftach Haitner, Thomas Holenstein, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy II: IE functions and universal one-way hashing. *Theory Comput.*, 16:1–55, 2020. 13, 14, 15, 16, 32, 44, 46, 47, 54

[HHRS15] Iftach Haitner, Jonathan J Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols—tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM Journal on Computing*, 44(1):193–242, 2015. 20

[HKAAL23] Jan Hendrik Kirchner, Lama Ahmad, Scott Aaronson, and Jan Leike. New AI classifier for indicating AI-written text. https://openai.com/blog/new-ai-classifier-for-indicating-ai-\written-text, 2023. Accessed: 2023-10-05. 20

[HLW+24] Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. Universally optimal watermarking schemes for LLMs: from theory to practice. arXiv:2410.02890 [cs.CR]., October 2024. 6, 20

[HLW+25] Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. Distributional information embedding: A framework for multi-bit watermarking. arXiv:2501.16558 [cs.CR]., January 2025. 6, 20

[HNO+09] Iftach Haitner, Minh Nguyen, Shien Jin Ong, Omer Reingold, and Salil Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009. 5, 12

[Hov16] Dirk Hovy. The enemy in your own camp: How well can we detect statistically-generated fake reviews–an adversarial study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 351–356, 2016. 2

[HPS08] Jeffrey Hoffstein, Jill Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 1 edition, 2008. 17

[HRVW09] Iftach Haitner, Omer Reingold, Salil Vadhan, and Hoeteck Wee. Inaccessible entropy. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2009. 5, 12, 24, 48

[HV17] Iftach Haitner and Salil Vadhan. The many entropies in one-way functions. In *Tutorials on the Foundations of Cryptography*, pages 159–217. Springer, 2017. 5, 12

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 12–24. ACM Press, 1989. 16

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 248–253, 1989. 16

[JAML20] Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks VS Lakshmanan. Automatic detection of machine generated text: A critical survey. *arXiv preprint arXiv:2011.01314*, 2020. 2

[KGW+23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. 2, 4, 5, 22

[KJGR21] Gabriel Kaptchuk, Tushar M. Jois, Matthew Green, and Aviel D. Rubin. Meteor: Cryptographically secure steganography for realistic distributions. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 1529–1548. ACM, 2021. 2

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Technical Report 2005/328, Cryptology ePrint Archive, 2005. 46, 53

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014. 10, 17

[KSK+23] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36, 2023. 2

[KTHL24] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. 2, 4

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979. 8, 10, 32

[LMC+24] Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, Xinyi (Alexis) Wu, Enrico Shippole, Kurt Bollacker, Tongshuang Wu, Luis Villa, Sandy Pentland, and Sara Hooker. A large-scale audit of dataset licensing and attribution in AI. *Nature Machine Intelligence*, 6(8):975–987, 2024. 2

[LPH+24] Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S. Yu. An unforgeable publicly verifiable watermark for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 3

[McE78] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978. 7

[Mer87] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378, 1987. 8

[MK05] Pierre Moulin and Ralf Koetter. Data-hiding codes. *Proceedings of the IEEE*, 93(12):2083–2126, December 2005. 6, 20

[MMP14] Mohammad Mahmoody, Hemanta K Maji, and Manoj Prabhakaran. Limits of random oracles in secure computation. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 23–34, 2014. 3

[MO03] Pierre Moulin and Joseph A. O'Sullivan. Information-theoretic analysis of information hiding. *IEEE Trans. Inf. Theory*, 49(3):563–593, 2003. 6

[Mou03] Pierre Moulin. Information-hiding games. In *Digital Watermarking*, pages 1–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 6, 20

[Mou05] Pierre Moulin. The method of types and its application to information hiding. In *2005 13th European Signal Processing Conference*, pages 1–1, 2005. 6, 20

[MS78] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978. 18, 19

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *2013 IEEE International Symposium on Information Theory*, pages 2069–2073, 2013. 7

[Nie86] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. 15(2):157–166, January 1986. 7

[Nis13] Ryo Nishimaki. How to watermark cryptographic functions. In *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2013. 7

[NSS99] David Naccache, Adi Shamir, and Julien P. Stern. How to copyright a function? In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, volume 1560 of *Lecture Notes in Computer Science*, pages 188–196. Springer, 1999. 7

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43, 1989. 17, 32, 52

[POC+23] Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche, editors, *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM, 2023. 8

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990. 10, 32, 46, 53

[RRR19] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. *SIAM Journal on Computing*, (0):16–255, 2019. 20

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 8

[Sen11] Nicolas Sendrier. *Niederreiter Encryption Scheme*, pages 842–843. Springer US, Boston, MA, 2011. 7

[Sha49a] C. E. Shannon. Communication theory of secrecy systems. *Bell Systen Technicl Journal*, 28:656–715, October 1949. 5

[Sha49b] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949. 5

[Sho00] Victor Shoup. A composition theorem for universal one-way hash functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 445–452, 2000. 52

[TCH23] Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. The science of detecting llm-generated texts. *arXiv preprint arXiv:2303.07205*, 2023. 2, 20

[TSZ19] Teik Guan Tan, Pawel Szalachowski, and Jianying Zhou. Challenges of post-quantum digital signing in real-world applications: A survey. Cryptology ePrint Archive, Paper 2019/1374, 2019. 8

[VKS20] Lav R. Varshney, Nitish Shirish Keskar, and Richard Socher. Limits of detecting text generated by large-scale language models. In *Proceedings of the Information Theory and Applications Workshop (ITA)*, February 2020. 2

[Yan15] Guang Yang. *Cryptography and Randomness Extraction in the Multi-Stream Model*. PhD thesis, Tsinghua University, Beijing, China, 2015. http://eccc.hpi-web.de/static/books/Cryptography_and_Randomness_Extraction_in_the_Multi_Stream_Model. 57

[ZALW24] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. 4, 20

[ZEF+] Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: impossibility of strong watermarking for language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. 4

[ZGC+24] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, Somesh Jha, Lei Li, Yu-Xiang Wang, and Dawn Song. Sok: Watermarking for ai-generated content, 2024. 4, 20

[ZHR+19] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *Advances in Neural Information Processing Systems*, 2019. 2

[ZWL24] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Watermarking for large language model. *Tutorials of ACL*, 2024. 4, 20

A Hash Error-Correcting Functions from One-Way Permutations

In this section, we derive a family of $\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$ functions based on the assumption of one-way permutations. These functions combine semantics of error-correcting codes and hash functions to ensure: (1) errors can be corrected during the process of embedding the message-signature pairs into the model output; (2) the chance of distinguishing the message-signature pairs from the model output is negligible in the security parameter.

First, we recall the definition of hash error-correcting functions.

Definition A.1 ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$)**.** *Let λ be the security parameter and $z \in \{0,1\}^\lambda$. Also, let $\boldsymbol{\sigma}$ be a signature, $\mathbf{m}$ be a message, and $\mathbf{c}$ be a codeword. We denote* ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$) *as* Hash Error-Correcting *functions that satisfy the following:*

- ***Error-Correcting**: $\mathsf{HashDecode}_z(\mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma}), \mathbf{m}) = \boldsymbol{\sigma}$ with probability at least $1 - \mathrm{neg}(\lambda)$.*
- ***Target Collision Resistance**: For every codeword $\mathbf{c}$ and message-signature pair $(\mathbf{m}, \boldsymbol{\sigma})$ such that $\mathbf{c} \leftarrow \mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma})$, only with probability at most $\mathrm{neg}(\lambda)$ can a PPT adversary find $(\mathbf{m}', \boldsymbol{\sigma}')$ such that $(\mathbf{m}, \boldsymbol{\sigma}) \neq (\mathbf{m}', \boldsymbol{\sigma}')$ and $\mathbf{c} = \mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma}) = \mathsf{HashEncode}_z(\mathbf{m}', \boldsymbol{\sigma}')$.*

The goal of hash error-correcting functions (Definition A.1) is to prevent an adversary from distinguishing message-signature pairs that will be embedded into generative model outputs. Distinguishing message-signature pairs would violate the distortion-free property (Definition 4.6).

We show how the existence of one-way permutations imply hash error-correcting functions via the use of inaccessible entropy. Along the way, several notions of entropy are quantified, manipulated, and amplified. We can rely on such operations to further manipulate the entropy of the tokens generated by the watermarking schemes.

See [HHR+20, HHR+10] for details—beyond signature-based watermarking schemes—on how one-way permutations imply inaccessible entropy generators which lead to hash error-correcting functions. We generally follow the approach of [HHR+20, HHR+10] in our proofs, with a few modifications and clarifications.

In the previous work of [HHR+20, HHR+10], there are two main constructions that yield universal one-way hash functions. For security parameter λ, one construction in [HHR+20, HHR+10] yields key and output length of $\tilde{O}(\lambda^{22})$ and the other (more efficient one) yields key and output length of $\tilde{O}(\lambda^7)$. Our construction yields key and output length of $\tilde{O}(\lambda^7)$.

In all the constructions, the procedure involves first obtaining a noticeable gap between real Shannon entropy and accessible max-entropy.

The following is the main theorem to be shown.

Theorem A.2. *Let λ be the security parameter. Suppose there exists a one-way permutation $f : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$. Then, there exists a family of* ($\mathsf{HashEncode}_z, \mathsf{HashDecode}_z$) *functions with codeword length $\widetilde{O}(\lambda^7)$. That is, for any $k \in \mathbb{N}$ such that $k \leq m(\lambda) < n(\lambda)$, there exists a family of functions*

$$\left\{\mathsf{HashEncode}_z : \{0,1\}^{n(\lambda)} \times \{0,1\}^k \mapsto \{0,1\}^{m(\lambda)}\right\}_{z \in \{0,1\}^\lambda},$$

$$\left\{\mathsf{HashDecode}_z : \{0,1\}^{m(\lambda)} \times \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^k\right\}_{z \in \{0,1\}^\lambda},$$

where $m(\lambda) = \widetilde{O}(\lambda^7)$ and satisfies the properties in Definition A.1.

Proof. We will construct the hash error-correcting functions of the form:

$$\mathsf{HashEncode}_z : \{0,1\}^{n(\lambda)} \times \{0,1\}^k \mapsto \{0,1\}^{m(\lambda)},$$

$$\mathsf{HashDecode}_z : \{0,1\}^{m(\lambda)} \times \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^k.$$

The family of hash error-correcting functions $(\mathsf{HashEncode}_z, \mathsf{HashDecode}_z)$ will follow the semantics of Definition A.1.

Let $m(\lambda) < n(\lambda)$ and define $\mathsf{H}_z : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ to be a target collision-resistant function. Recall that the property of target collision resistance guarantees that the probability that a PPT adversary A succeeds in the following game is negligible in λ:

i. Let $(x, \mathsf{state}) \xleftarrow{\mathrm{R}} \mathsf{A}(1^\lambda)$.

- Abort if $(x, \mathsf{state}) \notin \{0,1\}^{n(\lambda)} \times \{0,1\}^*$.

ii. Let $z \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$.

iii. Let $x' \xleftarrow{\mathrm{R}} \mathsf{A}(\mathsf{state}, z)$.

- Abort if $x' \notin \{0,1\}^{n(\lambda)}$.

iv. A *succeeds* if $x \neq x'$ and $\mathsf{H}_z(x) = \mathsf{H}_z(x')$.

For some $k \leq m(\lambda) - d + 1$, let $(\mathsf{Encode}, \mathsf{Decode})$ be any $[m(\lambda), k, d]$ error-correcting code (Definition 3.30) of the form:

$$\mathsf{Encode} : \{0,1\}^k \mapsto \{0,1\}^{m(\lambda)},$$

$$\mathsf{Decode} : \{0,1\}^{m(\lambda)} \mapsto \{0,1\}^k.$$

Such codes exist by Theorem 3.34 and Theorem 3.35.

Now we can define $\mathsf{HashEncode}_z$ in terms of H_z and Encode functions.

$$\begin{aligned} \mathbf{c} &\leftarrow \mathsf{HashEncode}_z(\mathbf{m}, \boldsymbol{\sigma}) && (2)\\ &= \mathsf{H}_z(\mathbf{m}) \oplus \mathsf{Encode}(\boldsymbol{\sigma}), && (3) \end{aligned}$$

Also, define $\mathsf{HashDecode}_z$ in terms of H_z and Decode functions.

$$\begin{aligned} \boldsymbol{\sigma} &\leftarrow \mathsf{HashDecode}_z(\mathbf{c}, \mathbf{m}) && (4)\\ &= \mathsf{Decode}(\mathsf{H}_z(\mathbf{m}) \oplus \mathbf{c}). && (5) \end{aligned}$$

Then for any $\mathbf{m} \in \{0,1\}^{n(\lambda)}$ and $\boldsymbol{\sigma} \in \{0,1\}^k$,

$$\begin{aligned} &\mathsf{Decode}(\mathsf{H}_z(\mathbf{m}) \oplus \mathsf{HashEncode}_z(\boldsymbol{\sigma}, \mathbf{m})) && (6)\\ &= \mathsf{Decode}(\mathsf{H}_z(\mathbf{m}) \oplus \mathsf{H}_z(\mathbf{m}) \oplus \mathsf{Encode}(\boldsymbol{\sigma})) && (7)\\ &= \mathsf{Decode}(\mathsf{Encode}(\boldsymbol{\sigma})) && (8)\\ &= \boldsymbol{\sigma}. && (9) \end{aligned}$$

This ensures the error-correcting property in Definition A.1, as long as the distance of the code is large enough to correct errors. (In the context of watermarking, we can show that with probability at least $1-\mathrm{neg}(\lambda)$, the codewords can be error-corrected.) Now, for the rest of the proof, we focus on proving the property of target collision-resistance. For $m(\lambda) < n(\lambda)$, let $\mathsf{H}_z : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ Clearly, the target collision-resistance of H_z would imply the target collision-resistance property in Definition A.1.

Fix $s := \log^2(\lambda)$. Then we use Theorem A.3 to get a function $\mathsf{F} : \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m$ with $\widetilde{\lambda} = O(\lambda)$ and gap $\Delta = \log\lambda/\lambda$ between real Shannon entropy and accessible average max-entropy.

Now by Theorem A.6 we get a family of hash error-correcting functions with output length $O(\widetilde{\lambda}^4 s/\Delta^3) = O(\lambda^7 \log^2(\lambda)/\log^3(\lambda)) = \widetilde{O}(\lambda^7)$, and key length

$$O(\widetilde{\lambda}^4 s/\Delta^3 \cdot \log(\lambda)) = \widetilde{O}(\lambda^7).$$

□

A.1 Inaccessible Entropy from One-Way Permutations

Recall that the accessible entropy of a function F is the entropy of the best *efficient* collision-finding algorithm F^{-1}. The inaccessible entropy of is the gap between its (real) entropy and its accessible entropy. The goal is to use the assumption of one-way permutations to obtain inaccessible entropy. This result is implied by Theorem A.3, which shows that a simplified variant of the some analysis in the work of Rompel [Rom90, KK05] yields inaccessible entropy with strong guarantees.

The function that is constructed is

$$\mathsf{F}(x,g,i) = (g(f(x))_{1,\ldots,i}, g),$$

where $g : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$ is a three-wise independent function. Recall that for any $\ell > 1$, a canonical example of an ℓ-wise independent function family is $\mathcal{H}_{s,t,\ell} = (\{0,1\}^s)^\ell$ defined by $(h_0,\ldots,h_{\ell-i})(x) := \sum_{0\le i\le \ell-1} h_i \cdot x^i$ over $\mathrm{GF}(2^s)$, truncated to its first t bits.

The benefits of the three-wise independence hashing step are that:

i. As shown in [HHR+20], obtaining a bound on accessible average max-entropy rather than accessible Shannon entropy could give better parameter settings. In particular, Theorem A.3 yields a bound on accessible average max-entropy rather than accessible Shannon entropy.

ii. As shown in [HHR+20], we get more inaccessible entropy ($\tilde{\Theta}(1/\lambda)$ bits rather than $\tilde{\Theta}(1/\lambda^2)$ bits).

These allow for a more efficient (and arguably simpler) transformation of F into functions with target collision-resistance.

Theorem A.3 (Inaccessible average max-entropy from one-way permutation)**.** *Let* $f\colon \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$ *be a one-way permutation and let*

$$\mathcal{G} = \left\{ g\colon \{0,1\}^\lambda \mapsto \{0,1\}^\lambda \right\}$$

be a family of constructible, three-wise independent hash functions. Define F *over* $\mathcal{D}(F) := \{0,1\}^\lambda \times \mathcal{G} \times [\lambda]$ *by*

$$\mathsf{F}(x,g,i) = (g(f(x))_{1,\ldots,i}, g, i).$$

Then F^{-1} *has accessible average max-entropy at most*

$$\mathrm{H}(Z \mid \mathsf{F}(Z)) - \Omega(\log \lambda/\lambda),$$

where Z *is uniformly distributed over* $\mathcal{D}(\mathsf{F})$.

Proof. The sets $\{\mathcal{L}(x,g,i)\}_{x\in\{0,1\}^\lambda, i\in[\lambda], g\in\mathcal{G}}$ realizing the inaccessible entropy of F^{-1} are defined by

$$\mathcal{L}(x,g,i) = \{(x',g,i)\colon f(x') \in \widetilde{\mathcal{L}}(f(x),i) \wedge g(f(x'))_{1,\ldots,i} = g(f(x))_{1,\ldots,i}\} \tag{10}$$

where for $y \in \{0,1\}^\lambda$ and $i \in [n]$, we let

$$\begin{aligned}\widetilde{\mathcal{L}}(y,i) &= \{y\} \cup \left\{y' \in \{0,1\}^\lambda\colon \underset{f(X)}{\mathrm{H}}(y') \geq (i + c\cdot\log\lambda)\right\} \\ &= \{y\} \cup \left\{y' \in \{0,1\}^\lambda\colon |f^{-1}(y')| \leq 2^{\lambda-i}/\lambda^c\right\}.\end{aligned} \tag{11}$$

where c is a sufficiently large constant. Namely, $\widetilde{\mathcal{L}}(y,i)$ consists, in addition to y itself, of "i-light" images with respect to f. Recall that the sample entropy is defined as

$$\underset{f(X)}{\mathrm{H}}(y) = \log(1/\mathbb{P}[f(X)=y]) = \lambda - \log\left|f^{-1}(y)\right|,$$

so the "heavy" images, where $f^{-1}(y)$ is large, have low sample entropy. As a warm-up, it is helpful to write down $\widetilde{\mathcal{L}}(y,i)$ and $\mathcal{L}(x,g,i)$ for the case where f is a one-way permutation. If f is a permutation, then $\widetilde{\mathcal{L}}(y,i)$ is given by:

$$\widetilde{\mathcal{L}}(y,i) = \begin{cases}\{0,1\}^\lambda & \text{if } i \leq \lambda - c\log\lambda \\ \{y\} & \text{otherwise.}\end{cases}$$

Then, for all $x \in \{0,1\}^\lambda$, we have $\mathbb{E}[|\mathcal{L}(x,G,i)|] = 2^{\lambda-i}$ for all $i \leq \lambda - c\log\lambda$ and $|\mathcal{L}(x,g,i)| = 1$ for all $g \in \mathcal{G}$ and all $i > \lambda - c\log\lambda$. This means that the entropy gap between $F^{-1}(F(Z))$ and $\mathcal{L}(X,G,I)$ is roughly

$$\frac{1}{\lambda}\sum_{i>\lambda-c\log\lambda} \lambda - i,$$

which is $\Omega(c^2\log^2\lambda/\lambda)$.

The proof of the theorem immediately follows by the following two claims.

Claim A.4 (Claim 4.6 in [HHR+20])**.** *For every* PPT F*-collision-finder* A *and every constant* $c > 0$*, it holds that*

$$\mathbb{P}[\mathsf{A}(Z;R) \notin \mathcal{L}(Z)] \leq \mathrm{neg}(\lambda),$$

where Z *is uniformly distributed over* $\mathcal{D}(F)$ *and* R *is uniformly distributed over the random coins of* A.

Claim A.5 (Claim 4.7 in [HHR+20])**.** *For any constant* c *it holds that*

$$\mathrm{E}[\log|\mathcal{L}(Z)|] \leq \mathrm{E}\left[\log\left|\mathsf{F}^{-1}(\mathsf{F}(Z))\right|\right] - \Omega\left(\frac{c\log\lambda}{\lambda}\right),$$

where Z *is uniformly distributed in* $\mathcal{D}(\mathsf{F})$.

□

A.2 Collision Resistance from Inaccessible Entropy

Here, we can show how to construct the collision-resistant function from any efficiently computable function with a noticeable gap between real Shannon entropy and either accessible average max-entropy or accessible Shannon entropy.

As done in previous work [HRVW09], we first transform the entropy gap into a noticeable gap between real Shannon entropy and accessible max-entropy. In this construction, we start from a gap between real Shannon entropy and accessible average max-entropy.

Theorem A.6. *Suppose there exists a polynomial-time computable function* $\mathsf{F} : \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m$ *such that* F^{-1} *has a noticeable gap* Δ *between real Shannon entropy and accessible average max-entropy. Then, there exists a family of collision-resistant hash functions with output length* $O(\widetilde{\lambda}^4 s/\Delta^3)$ *and key length* $O(\widetilde{\lambda}^4 s/\Delta^3 \cdot \log \lambda)$ *for any* $s = \omega(\log \lambda)$*, where* λ *is the security parameter.*

Overview. To prove Theorem A.6, the construction proceeds via a series of transformations: gap amplification (via repetition), entropy reduction (by hashing inputs), and reducing output length (by hashing outputs). In some of these transformations, λ_0 denotes the input length of the function F we start with, and λ to denote the security parameter.

A.2.1 Gap Amplification

Here, we show a standard result that the direct product construction increases the gap between real entropy and accessible entropy. As noted by the previous work of [HRVW09], another useful effect of direct product (for certain settings of parameters) is turning real Shannon entropy into real min-entropy, and turning accessible average max-entropy into accessible max-entropy.

Lemma A.7 (Gap amplification)**.** *Let* λ *be a security parameter and* $\widetilde{\lambda} = O(\lambda)$*. Define the function* $\mathsf{F}\colon \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m$*. For* $t \in \mathrm{poly}(\lambda)$*, let* F^t *be the* t*-fold direct product of* F*. Then,* F^t *satisfies the following properties:*

i. If F^{-1} *has real Shannon entropy at least* k*, then* $(\mathsf{F}^t)^{-1}$ *has real min-entropy at least* $t \cdot k - \widetilde{\lambda} \cdot \sqrt{st}$ *for any* $s = \omega(\log \lambda)$ *and* $t > s$*.*

ii. If F^{-1} *has accessible average max-entropy at most* k*, then* $(\mathsf{F}^t)^{-1}$ *has accessible max-entropy at most* $t \cdot k + \widetilde{\lambda} \cdot \sqrt{st}$ *for any* $s = \omega(\log \lambda)$*.*

Proof. In the following X and $X^{(t)} = (X_1, \ldots, X_t)$ are uniformly distributed over $\{0,1\}^{\widetilde{\lambda}}$ and $(\{0,1\}^{\widetilde{\lambda}})^t$, respectively.

i. Follows readily from appling Lemma C.5 with $\epsilon = 2^{-s}$.

ii. Given any PPT F^t-collision-finder A', we construct a PPT F-collision-finder A that:

> On input x, picks a random i in $[t]$ along with random $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_t$, computes $\mathsf{A}'(x_1, \ldots, x_t) \mapsto (x'_1, \ldots, x'_t)$, and outputs x'_i.

By the bound on the accessible average max-entropy of F^{-1}, we know that there exists a family of sets $\{\mathcal{L}(x)\}$ such that $\mathbb{E}\big[\log|\mathcal{L}(X)|\big] \leq k$, $x \in \mathcal{L}(x)$, and $\mathbb{P}[\mathsf{A}(X) \notin \mathcal{L}(X)] \leq \mathrm{neg}(\lambda)$. Consider the family of sets $\left\{\mathcal{L}'(x^{(t)})\colon x^{(t)} \in (\{0,1\}^{\widetilde{\lambda}})^t\right\}$ given by:

$$\mathcal{L}'(x^{(t)}) = \mathcal{L}(x_1^{(t)}) \times \mathcal{L}(x_2^{(t)}) \times \cdots \times \mathcal{L}(x_t^{(t)}).$$

By linearity of expectations, we have $\mathbb{E}\big[\log|\mathcal{L}'(X_1,\ldots,X_t)|\big] \leq t{\cdot}k$. Moreover, by the Chernoff-Hoeffding bound and using the fact that $\log|\mathcal{L}(X)|$ assumes values in $[0,\widetilde{\lambda}]$, we have

$$\begin{aligned}&\mathbb{P}\Big[\log\Big|\mathcal{L}'(X^{(t)})\Big| \geq t\cdot k + \widetilde{\lambda}\sqrt{st}\Big] \\ &= \mathbb{P}\Big[\log\Big|\mathcal{L}(X_1^{(t)})\Big| + \cdots + \log\Big|\mathcal{L}(X_t^{(t)})\Big| \geq t\cdot k + \widetilde{\lambda}\sqrt{st}\Big] \leq e^{-2s}.\end{aligned} \tag{12}$$

We claim that this implies that A' has accessible max-entropy at most $t\cdot k + \widetilde{\lambda}\sqrt{st}$. Suppose otherwise, then there exists a non-negligible function ϵ such that

$$\mathbb{P}[\mathsf{A}'(\mathsf{F}^t(X^{(t)})) \notin \mathcal{L}'(X^{(t)})] \geq \epsilon - e^{-2s} \geq \epsilon/2$$

Therefore,

$$\mathbb{P}[\mathsf{A}(\mathsf{F}(X)) \notin \mathcal{L}(X)] = \mathbb{P}[\mathsf{A}'(\mathsf{F}^t(X^{(t)})) \notin \mathcal{L}'(X^{(t)})]/t \geq \epsilon/2t$$

which contradicts our assumption on A.

□

A.2.2 Entropy Reduction

Next, consider a construction that, given F and any parameter ℓ, reduces the accessible max-entropy of F^{-1} by roughly ℓ bits. At the same time, the construction (approximately) preserves the gap between real min-entropy and accessible max-entropy.

Lemma A.8 (Reducing entropy)**.** *Let λ be a security parameter and define $\widetilde{\lambda} = O(\lambda)$. Fix the function $\mathsf{F}\colon \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m$. Consider the family of pairwise independent hash functions $\mathcal{G} = \left\{g\colon \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^\ell\right\}$. Then,*

$$\mathsf{F}'\colon \{0,1\}^{\widetilde{\lambda}} \times \mathcal{G} \mapsto \{0,1\}^m \times \mathcal{G} \times \{0,1\}^\ell,$$

as defined by $\mathsf{F}'(x,g) = (\mathsf{F}(x), g, g(x))$ satisfies the following properties:

i. Assuming F^{-1} has real min-entropy at least k, then $(\mathsf{F}')^{-1}$ has real min-entropy at least $k-\ell-s$ for any $s = \omega(\log\lambda)$.

ii. Assuming F^{-1} has accessible max-entropy at most k, then $(\mathsf{F}')^{-1}$ has accessible max-entropy at most $\max\{k-\ell+s, 0\}$ for any $s = \omega(\log\lambda)$.

Proof. Let X be uniformly distributed over $\{0,1\}^{\widetilde{\lambda}}$ and G be uniformly distributed over the set of functions $\mathcal{G}$.

i. Fix $g \in \mathcal{G}$ and let $S_g = \left\{ z \in \{0,1\}^\ell \colon \mathbb{P}[g(X) = z] \le 2^{-\ell - s} \right\}$. Observe the following.

(a) $\mathbb{P}[g(X) \in S_g] \le 2^{-s}$ (by a union bound over $z \in S_g$);

(b) Fix any $z \notin S_g$ and any $x \in \{0,1\}^\lambda$ such that $\mathrm{H}_{X|F(X)}(x \mid F(x)) \ge k$. Then,

$$\begin{aligned}\mathbb{P}[X = x \mid \mathsf{F}'(X, g) = (\mathsf{F}(x), g, z)] &= \mathbb{P}[X = x \mid \mathsf{F}(X) = \mathsf{F}(x) \wedge g(X) = z] \\ &\le \frac{\mathbb{P}[X = x \mid \mathsf{F}(X) = F(x)]}{\mathbb{P}[g(X) = z]} \\ &\le \frac{2^{-k}}{2^{-\ell - s}} = 2^{-(k - \ell - s)}.\end{aligned}$$

where the second inequality follows from our assumptions on z and x.

Combining the above two observations and the bound on the real min-entropy of F, it follows that for all $g \in \mathcal{G}$, with probability $1 - 2^{-s} - \mathrm{neg}(\lambda)$ over $x \xleftarrow{\mathrm{R}} X$, we have

$$\mathbb{P}[X = x \mid \mathsf{F}'(X, g) = \mathsf{F}'(x, g)] \le 2^{-(k - \ell - s)}.$$

The bound on the real min-entropy of F' follows readily.

ii. Given a PPT F'-collision-finder A', we construct a PPT F-collision-finder A as follows:

On input x, picks a pair (g, r) uniformly at random and output $\mathsf{A}'(x, g; r)$.

By the bound on the accessible max-entropy of F^{-1}, we know that there exists a family of sets $\left\{ \mathcal{L}(x) \subseteq \{0,1\}^{\widetilde{\lambda}} \colon x \in \{0,1\}^{\widetilde{\lambda}} \right\}$ such that $|\mathcal{L}(x)| \le 2^k$, $x \in \mathcal{L}(x)$, and

$$\mathbb{P}\big[\mathsf{A}(X, \mathrm{G}; R) \in \mathcal{L}(X)\big] \ge 1 - \mathrm{neg}(\lambda), \tag{13}$$

where R is uniformly distributed over the random coins of A.

Let $\mathcal{L}'(x, g) := \{(x', g) \colon x' \in \mathcal{L}(x) \wedge g(x') = g(x)\}$. Equation (13) yields that

$$\mathbb{P}\big[\mathsf{A}'(X, \mathrm{G}; R) \in \mathcal{L}'(X, \mathrm{G})\big] \ge 1 - \mathrm{neg}(\lambda). \tag{14}$$

We next bound the size of the set $\mathcal{L}'(x, g)$. Fix any $x \in \{0,1\}^\lambda$. For any $x' \ne x$, pairwise independence of $\mathcal{G}$ tells us that $\mathbb{P}[G(x') = G(x)] = 2^{-\ell}$. It follows from linearity of expectation that

$$\mathrm{E}\big[\big|\mathcal{L}'(x, \mathrm{G}) \setminus \{x\}\big|\big] \le |\mathcal{L}(x)| \cdot 2^{-\ell} \le 2^{k - \ell}.$$

Then, by Markov's inequality, we have

$$\mathbb{P}\big[\big|\mathcal{L}'(x, \mathrm{G})\big| \le 2^{k - \ell + s - 1} + 1\}\big] \ge 1 - 2^{-(s-1)}, \tag{15}$$

Combining the last two inequalities, we obtain

$$\mathbb{P}\big[\mathsf{A}'(X, \mathrm{G}; R) \in \mathcal{L}'(X, \mathrm{G}) \wedge \big|\mathcal{L}'(X, \mathrm{G})\big| \le \max\left\{2^{k - \ell + s}, 1\right\}\big] \ge 1 - \mathrm{neg}(\lambda) - 2^{-(s-1)}. \tag{16}$$

This yields an upper bound of $\max\{k - \ell + s, 0\}$ on the accessible max-entropy of $(\mathsf{F}')^{-1}$.

□

A.2.3 Reducing Output Length

In Lemma A.9, a transformation is given that derives a function that is both length-decreasing and collision-resistant on random inputs.

Lemma A.9 (Reducing output length)**.** *Let λ be a security parameter and $\widetilde{\lambda} = O(\lambda)$. Then define the function $\mathsf{F}\colon \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m$. Fix a family of pairwise independent hash functions $\mathcal{G} = \left\{g\colon \{0,1\}^m \mapsto \{0,1\}^{\widetilde{\lambda}-\log\lambda}\right\}$ and let $\mathsf{F}'\colon \mathcal{G}\times\{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^{\widetilde{\lambda}-\log\lambda}\times\mathcal{G}$ be defined by $\mathsf{F}'(x,g) = (g, g(\mathsf{F}(x)))$. The following holds: if F^{-1} has real min-entropy at least $\omega(\log\lambda)$ and F is collision-resistant on random inputs, then F' is collision-resistant on random inputs.*

Proof. Let $\mathrm{Im}(\mathsf{F})$ be the image of the function F.

The bound on real min-entropy implies there exists a subset $S \subseteq \{0,1\}^{\widetilde{\lambda}}$ of density at most $\mathrm{neg}(\lambda)$, such that for all $x \notin S$ it holds that $\left|\mathsf{F}^{-1}(\mathsf{F}(x))\right| = \lambda^{\omega(1)}$. Hence,

$$|\mathrm{Im}(\mathsf{F})| \leq |\mathsf{F}(S)| + \left|\mathsf{F}(\bar{S})\right| \leq |S| + \left|\bar{S}\right|/\lambda^{\omega(1)} \leq \mathrm{neg}(\lambda)\cdot 2^{\lambda}. \tag{17}$$

By the two-wise independence of $\mathcal{G}$,

$$\mathbb{P}\left[\exists y' \in \mathrm{Im}(\mathsf{F})\colon y' \neq \mathsf{F}(X) \wedge \mathrm{G}(y') = \mathrm{G}(\mathsf{F}(X))\right] \leq \frac{|\mathrm{Im}(\mathsf{F})|}{2^{\widetilde{\lambda}-\log\lambda}} \leq \mathrm{neg}(\lambda). \tag{18}$$

Namely, $g(\mathsf{F}(x))$ uniquely determines $\mathsf{F}(x)$ with high probability. In particular, a collision for $g\circ\mathsf{F}$ is also a collision for F. Given any PPT F'-collision-finder A', we construct a PPT F-collision-finder A as follows:

> On input x, pick g and r at random and compute $x' = \mathsf{A}'(x, g; r)$. If $\mathsf{F}(x') = \mathsf{F}(x)$, output x', else output x.

Equation (18) implies that $\mathbb{P}[\mathsf{A}'(X,\mathrm{G};R) \neq (\mathsf{A}(X;\mathrm{G},R),\mathrm{G})] \leq \mathrm{neg}(\lambda)$. Therefore, $\mathbb{P}[\mathsf{A}'(X,\mathrm{G};R) = (X,\mathrm{G})] \geq 1-\mathrm{neg}(\lambda)$. Namely, F' is also collision-resistant on random inputs. □

A.2.4 Additional Transformations

Next are two more standard transformations that can be used to complete the construction.

Lemma A.10 (From random inputs to targets, folklore)**.** *Let λ be a security parameter and $\widetilde{\lambda} = O(\lambda)$. Define the length-decreasing function $\mathsf{F}\colon \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m$. Suppose F is collision-resistant on random inputs. Then, $\left\{\mathsf{F}'_y\colon \{0,1\}^{\widetilde{\lambda}} \mapsto \{0,1\}^m\right\}_{y\in\{0,1\}^{\widetilde{\lambda}}}$ as defined by $\mathsf{F}'_y(x) = \mathsf{F}(y+x)$ is a family of target collision-resistant hash functions.*

Proof. Let A' be a PPT adversary that breaks target collision-resistance of F'_y, we can construct a PPT adversary A that breaks F as follows:

> On input x, run $\mathsf{A}'(1^\lambda)$ to compute (x_0, state), and then run $\mathsf{A}'(\mathsf{state}, x\oplus x_0)$ to compute x_1. Output $x \oplus x_0 \oplus x_1$.

Note that (x_0, x_1) is a collision for $\mathsf{F}'_{x\oplus x_0}$ iff $(x, x\oplus x_0\oplus x_1)$ is a collision for F. It then follows quite readily that A breaks F with the same probability that A' breaks F'_y. □

The following result is from [Sho00] and improves on the work of [NY89, BR97]. The result implies that we can construct target collision-resistant hash functions for arbitrarily long inputs starting from one for a fixed input length.

Lemma A.11 (Increasing the input length [Sho00])**.** *Let λ be a security parameter, $t = \mathrm{poly}(\lambda)$ be a parameter and let $\left\{\mathsf{F}_y \colon \{0,1\}^{\widetilde{\lambda}+\log\lambda} \mapsto \{0,1\}^{\widetilde{\lambda}}\right\}$ be a family of target collision-resistant hash functions. Then, there exists a family of target collision-resistant hash functions $\left\{\mathsf{F}'_{y'} \colon \{0,1\}^{\widetilde{\lambda}+t\log\lambda} \mapsto \{0,1\}^{\widetilde{\lambda}}\right\}$ where $|y'| = O(|y|\log t)$.*

A.2.5 Putting Everything Together

Now using the previous transformations, one can prove Theorem A.6:

Proof of Theorem A.6. The security parameter is λ and $\lambda_0 = O(\lambda)$.

Below the entropy gap is Δ and the parameter s is set to $s \in \omega(\log\lambda)$.

Recall that we have given $\mathsf{F} : \{0,1\}^{\lambda_0} \mapsto \{0,1\}^{m_0}$,

STEP 1 **(gap amplification):** For a parameter t, we define F_1 as $\mathsf{F}_1(x_1,\ldots,x_t) = (\mathsf{F}(x_1),\ldots,\mathsf{F}(x_t))$, the t-fold direct product of F. We choose the parameter $t \in O(\lambda_0^2 s/\Delta^2)$ such that

$$t \cdot k_{\mathrm{REAL}} - \lambda_0 \cdot \sqrt{st} \geq t \cdot (k_{\mathrm{REAL}} - \Delta/2) + \lambda_0 \cdot \sqrt{st} + 3s.$$

Lemma A.7 yields that this repetition increases both the real and accessible entropies of F_1 by a factor of t (compared to F). In addition, this repetition converts real Shannon entropy to real min-entropy and accessible average max-entropy to accessible max-entropy (up to additive terms that are sublinear in t). More precisely, we have the following properties:

- $\mathsf{F}_1 : \{0,1\}^{\lambda_1} \mapsto \{0,1\}^{m_1}$, where $\lambda_1(\lambda) = t \cdot \lambda_0$ and $m_1(\lambda) = t \cdot m_0$.
- F_1^{-1} has real min-entropy at least $t \cdot k_{\mathrm{REAL}} - \lambda_0 \cdot \sqrt{st} \geq t \cdot (k_{\mathrm{REAL}} - \Delta/2) + \lambda_0 \cdot \sqrt{st} + 3s$.
- F_1^{-1} has accessible max-entropy at most $t \cdot (k_{\mathrm{REAL}} - \Delta) + \lambda_0 \cdot \sqrt{st}$.

In steps 2 to 4, the construction uses non-uniform advice k, which corresponds to an approximation to k_{REAL}. In step 5, we will remove this non-uniform advice via "exhaustive search". Concretely, for steps 2 to 4, we are given k satisfying

$$k \in [k_{\mathrm{REAL}}, k_{\mathrm{REAL}} + \Delta/2] \tag{19}$$

This means that

- F_1^{-1} has real min-entropy at least $t \cdot (k - \Delta) + \lambda_0 \cdot \sqrt{st} + 3s$.
- F_1^{-1} has accessible max-entropy at most $t \cdot (k - \Delta) + \lambda_0 \cdot \sqrt{st}$.

This yields a gap of $3s$ between real min-entropy and accessible max-entropy.

STEP 2 **(entropy reduction):** We next apply entropy reduction to F_1 to obtain $\mathsf{F}_2^{(k)}$. That is, $\mathsf{F}_2^{(k)}(x,g) = (\mathsf{F}_1(x), g, g(x))$, where $g \colon \{0,1\}^{\lambda_1} \mapsto \{0,1\}^{\ell}$ is selected from a family of pairwise independent hash functions with $\ell = t \cdot (k - \Delta) + \lambda_0 \cdot \sqrt{st} + s = O(t\lambda_0)$. Lemma A.8 yields that this additional hashing reduces the real min-entropy and accessible max-entropy by ℓ (up to an additive term of s). More exactly, we have the following properties:

- $\mathsf{F}_2^{(k)}\colon \{0,1\}^{\lambda_2} \mapsto \{0,1\}^{m_2}$ where $\lambda_2(\lambda,k) = O(t\lambda_0)$ and $m_2(\lambda,k) = O(t\lambda_0)$. Note that in particular λ_2 and m_2 also depend on k (unlike λ_1 and m_1).
- If (19) holds, then $(\mathsf{F}_2^{(k)})^{-1}$ has real min-entropy at least s.
- If (19) holds, then $(\mathsf{F}_2^{(k)})^{-1}$ has accessible max-entropy at most 0. Hence, $\mathsf{F}_2^{(k)}$ is collision-resistant on random inputs (by Lemma 3.19).

STEP 3 **(reducing the output length):** We next reduce the output length of $\mathsf{F}_2^{(k)}$ by hashing the output to $\lambda_2 - \log\lambda$ bits. That is, $\mathsf{F}_3^{(k)}(x,g) = (g, g(\mathsf{F}_2^{(k)}(x)))$ where $g\colon \{0,1\}^{m_2} \mapsto \{0,1\}^{\lambda_2 - \log\lambda}$ is selected from a family of pairwise-independent hash functions.

- $\mathsf{F}_3^{(k)}\colon \{0,1\}^{\lambda_3} \mapsto \{0,1\}^{m_3}$ where $\lambda_3(\lambda,k) = O(t\lambda_0)$ and $m_3(\lambda,k) = \lambda_3 - \log\lambda$.
- By Lemma A.9, $\mathsf{F}_3^{(k)}$ is collision-resistant on random inputs, assuming that (19) holds.

STEP 4 **(adding random shifts)** We then transform $\mathsf{F}_3^{(k)}$ into a family $\left\{G_y^{(k)}\right\}$ of target collision-resistant hash functions via a random shift, following Lemma A.10. That is, $G_y^{(k)}(x) = \mathsf{F}_3^{(k)}(y+x)$. We then have that

- $G_y^{(k)}(x) : \{0,1\}^{\lambda_3} \mapsto \{0,1\}^{m_3}$ and $G_y^{(k)}$ uses a key y of length $\lambda_3(\lambda,k)$.
- If (19) holds, then $\left\{G_y^{(k)}\right\}$ is target collision-resistant.

STEP 5 **(removing non-uniformity):** To remove the non-uniform advice k, we do a brute-force search from 0 to λ_0 in steps of size $\Delta/2$, similar to the approach used in [Rom90] (see also [KK05, Section 3.6]).

i. First, we construct $\kappa = \lambda_0 \cdot 2/\Delta$ families of functions $\left\{G_y^{(k)}\right\}$, where we instantiate $\left\{G_y^{(k)}\right\}$ for all $k \in \left\{\frac{\Delta}{2}, 2\cdot\frac{\Delta}{2}, 3\cdot\frac{\Delta}{2}, \ldots, \lambda_0\right\}$. These κ families of functions satisfy the following properties:
   - Each of $G_y^{(k)}$ is length-decreasing; in particular, $G_y^{(k)}$ has input length $\lambda_3(\lambda,k)$ and output length $\lambda_3(\lambda,k) - \log\lambda$. Note that $G_y^{(\lambda_0)}$ has the longest input length, i.e., $\lambda_3(\lambda, i\Delta/2) \le \lambda_3(\lambda,\lambda_0)$ for all i because $\ell(\lambda,k)$ increases as a function of k. We may then assume that all κ functions $G_y^1, \ldots, G_y^\kappa$ have the same input length $\lambda_3(\lambda,\lambda_0)$ and the same output length $\lambda_3(\lambda,\lambda_0) - \log\lambda$ by padding the "extra part" of the input to the output.
   - At least one of the $\left\{G_y^{(k)}\right\}$ is target collision-resistant; this is because $k_{\text{REAL}} \in [0,\lambda_0]$, and so (19) holds for some k which we picked.

ii. Next, for each k, we construct a family of functions $\left\{\tilde{G}_{\tilde{y}}^{(k)}\right\}$ from $\left\{G_y^{(k)}\right\}$ with input length $\kappa\cdot\lambda_3(\lambda,\lambda_0)$, key length $O(\lambda_3(\lambda,\lambda_0)\cdot\log\lambda)$, and output length $\lambda_3(\lambda,\lambda_0) - \log\lambda$, by following the construction given by Lemma A.11. Again, at least one of the $\left\{\tilde{G}_{\tilde{y}}^{(k)}\right\}$ for k as above is target collision-resistant.

iii. Finally, we define a family of functions $\{G_{\tilde{y}_1,\ldots,\tilde{y}_\kappa}\}$ to be the concatenation of all $\tilde{G}_{\tilde{y}}^{(k)}$ on the same input. That is, $G_{\tilde{y}_1,\ldots,\tilde{y}_\kappa}(x) = \tilde{G}_{\tilde{y}_1}^{(\Delta/2)}(x) \circ \cdots \circ \tilde{G}_{\tilde{y}_\kappa}^{(n_0)}(x)$.

- Note that G has input length $\kappa \cdot \lambda_3(\lambda, \lambda_0)$ and output length $\kappa \cdot (\lambda_3(\lambda, \lambda_0) - \log \lambda)$, so G is length-decreasing.
- Moreover, since at least one of $\left\{\tilde{G}_{\tilde{y}_1}^{(\Delta/2)}(x)\right\}, \ldots, \left\{\tilde{G}_{\tilde{y}_\kappa}^{(n_0)}\right\}$ is target collision-resistant, $\{G_{\tilde{y}_1,\ldots,\tilde{y}_\kappa}\}$ must also be target collision-resistant. This is because a collision for $G_{\tilde{y}_1,\ldots,\tilde{y}_\kappa}$ is a collision for each of $\tilde{G}_{\tilde{y}_1}^{(\Delta/2)}, \ldots, \tilde{G}_{\tilde{y}_\kappa}^{(n_0)}$.

The family $\{G_{\tilde{y}_1,\ldots,\tilde{y}_\kappa}\}$ is the hash error-correcting function we wanted to construct, and so this finishes the proof of Theorem A.6. □

B Universal One-Way Hash Functions from One-Way Permutations

Let us recall the definition of a **Universal One-Way Function**.

For universal one-way hash function family, $\mathcal{F}_\lambda = \left\{\mathsf{F}_z \colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^{m(\lambda)}\right\}$, a function $\mathsf{F}_z \colon \{0,1\}^{n(\lambda)} \mapsto \{0,1\}^{m(\lambda)}$ has *input length* $n(\lambda)$, *key length* λ, and *output length* $m(\lambda)$.

A function F_z is a **Universal One-Way Function** (UOWF) if:

i. **Efficiently Computable:** Given an input x, it is easy to compute $\mathsf{F}_z(x)$.

ii. **Target Collision Resistance:** Given a randomly chosen x, it is computationally hard to find *any* different x' such that $\mathsf{F}_z(x) = \mathsf{F}_z(x')$.

Unlike standard **one-way functions** (OWFs), which require that it is hard to invert $\mathsf{F}_z(x)$ for a randomly chosen output y, UOWFs only require that after seeing x, an adversary cannot find a different input x' that maps to the same output.

Here are some key differences between universal one-way functions and one-way functions:

Property	**One-Way Function (OWF)**	**Universal One-Way Function (UOWF)**
Hardness	Hard to invert $f(x)$	Hard to find a second preimage x' for random x
Input Selection	Adversary picks y and tries to invert	Adversary sees x first and must find $x' \neq x$
Strength	Stronger security assumption	Weaker, but still useful

Haitner et al. [HHR+20, HHR+10] show how inaccessible entropy generators lead to universal one-way hash functions (UOWHF). One of their main results is the following:

Theorem B.1 (See [HHR+20, HHR+10])**.** *Suppose there exists a one-way permutation $f : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$. Then, there exists a family of universal one-way hash functions with key and output length $\widetilde{O}(\lambda^{22})$.*

And the other is:

Theorem B.2 (See [HHR+20, HHR+10])**.** *Suppose there exists a one-way permutation $f : \{0,1\}^\lambda \mapsto \{0,1\}^\lambda$. Then, there exists a family of universal one-way hash functions with key and output length $\widetilde{O}(\lambda^{7})$.*

C Relations Between Entropy Notions

In this section, we state and prove some (known) statements about how the different entropic notions (i.e., $\mathrm{H}_\infty(X), \mathrm{H}_2(X), \mathrm{H}(X), \mathrm{H}_0(X)$) relate. For completeness, we show the proofs of the relations. However, these are all standard information-thoeretic manipulations.

The following fact quantifies the probability that the sample-entropy is larger than the max-entropy.

Lemma C.1. *For random variable X it holds that*

i. $\mathrm{E}_{x \xleftarrow{R} X}\left[2^{\mathrm{H}_X(x)}\right] = |\mathrm{Supp}(X)|$.

ii. $\mathbb{P}_{x \xleftarrow{R} X}\left[\mathrm{H}_X(x) > \log \frac{1}{\varepsilon} + \mathrm{H}_0(X)\right] < \varepsilon$, *for any* $\varepsilon > 0$.

Proof. For the first item, compute

$$\begin{aligned}\mathrm{E}_{x \xleftarrow{R} X}\left[2^{\mathrm{H}_X(x)}\right] &= \sum_{x \in \mathrm{Supp}(X)} 2^{-\mathrm{H}_X(x)} \cdot 2^{\mathrm{H}_X(x)} \\ &= \sum_{x \in \mathrm{Supp}(X)} 1 \\ &= |\mathrm{Supp}(X)|.\end{aligned}$$

The second item follows by the first item and Markov's inequality.

$$\begin{aligned}\mathbb{P}_{x \xleftarrow{R} X}\left[\underset{X}{\mathrm{H}}(x) > \log \frac{1}{\varepsilon} + \mathrm{H}_0(X)\right] &= \mathbb{P}_{x \xleftarrow{R} X}\left[2^{\mathrm{H}_X(x)} > \frac{1}{\varepsilon} \cdot |\mathrm{Supp}(X)|\right] \\ &< \varepsilon.\end{aligned}$$

□

The following fact quantifies the contribution of unlikely events with high sample-entropy to the overall entropy of a random variable.

Lemma C.2. *Let X be a random variable with $\mathbb{P}_{x \xleftarrow{R} X}[\mathrm{H}_X(x) > k] \leq \varepsilon \in [0,1]$. Then $\mathrm{H}(X) \leq (1-\varepsilon)k + \varepsilon \cdot (\mathrm{H}_0(X) - \log \varepsilon) \leq k + \varepsilon \cdot \mathrm{H}_0(X) + 1$.*

Proof. Let $\mathcal{Y} = \{x \in \mathrm{Supp}(X) \colon \mathrm{H}_X(x) > k\}$ and let $Y = X|_{X \in \mathcal{Y}}$. Note that

$$\mathrm{H}(Y) = \mathrm{E}_{y \leftarrow Y}\left[\underset{Y}{\mathrm{H}}(y)\right] = \mathrm{E}_{y \leftarrow Y}\left[\underset{X}{\mathrm{H}}(y)\right] + \log \mathbb{P}_{x \xleftarrow{R} X}\left[\underset{X}{\mathrm{H}}(x) > k\right]. \tag{20}$$

Let $\varepsilon' = \mathbb{P}_{x \xleftarrow{R} X}[\mathrm{H}_X(x) > k]$ (hence, $\varepsilon' \leq \varepsilon$). We conclude that

$$\begin{aligned}\mathrm{H}(X) &= \sum_{x \in \mathrm{Supp}(X) \setminus \mathcal{Y}} \mathbb{P}[X = x] \cdot \underset{X}{\mathrm{H}}(x) + \sum_{x \in \mathcal{Y}} \mathbb{P}[X = x] \cdot \underset{X}{\mathrm{H}}(x) \\ &\leq (1-\varepsilon')k + \varepsilon' \cdot \sum_{x \in \mathcal{Y}} \mathbb{P}[Y = x] \cdot \underset{X}{\mathrm{H}}(x) \\ &= (1-\varepsilon')k + \varepsilon' \cdot (\mathrm{H}(Y) - \log \varepsilon') \\ &\leq (1-\varepsilon')k + \varepsilon' \cdot (\mathrm{H}_0(X) - \log \varepsilon') \\ &\leq (1-\varepsilon)k + \varepsilon \cdot (\mathrm{H}_0(X) - \log \varepsilon) \\ &\leq k + \varepsilon \cdot \mathrm{H}_0(X) + 1.\end{aligned}$$

□

Conditional entropies. We will also be interested in conditional versions of entropy. For jointly distributed random variables (X, Y) and $(x, y) \in \mathrm{Supp}(X, Y)$, we define the *conditional sample-entropy* to be

$$\underset{X|Y}{\mathrm{H}}(x|y) = \log \frac{1}{\mathbb{P}_{X|Y}[x|y]} = \log \frac{1}{\mathbb{P}[X=x|Y=y]}.$$

Then the standard *conditional Shannon entropy* can be written as

$$\mathrm{H}(X \mid Y) = \mathop{\mathbb{E}}_{(x,y)\overset{\mathrm{R}}{\leftarrow}(X,Y)}\left[\underset{X|Y}{\mathrm{H}}(x \mid y)\right] \tag{21}$$

$$= \mathop{\mathbb{E}}_{y\overset{\mathrm{R}}{\leftarrow}Y}\left[\mathrm{H}(X|_{Y=y})\right] \tag{22}$$

$$= \mathrm{H}(X, Y) - \mathrm{H}(Y). \tag{23}$$

The following known lemma states that conditioning on a variable with small support is unlikely to change the sample-entropy significantly. The result below can be further improved by considering the *average* sample-entropy induced by the conditioning, and in particular the *average min-entropy* of the variable [DORS08]. However, it could only improve by results by a constant. So we state the simpler statement below.

Lemma C.3. *Let X and Y be random variables, let $k = \mathrm{H}_\infty(X)$, and let $\ell = \mathrm{H}_0(Y)$. Then, for any $t > 0$, it holds that*

$$\mathbb{P}_{(x,y)\overset{R}{\leftarrow}(X,Y)}\left[\underset{X|Y}{\mathrm{H}}(x|y) < k - \ell - t\right] < 2^{-t}.$$

Proof. For $y \in \mathrm{Supp}(Y)$, let

$$\mathcal{X}_y = \left\{x \in \mathrm{Supp}(X) \colon \underset{X|Y}{\mathrm{H}}(x|y) < k - \ell - t\right\}.$$

We have $|\mathcal{X}_y| < 2^{k-\ell-t}$. Hence,

$$\left|\mathcal{X} = \bigcup_{y\in\mathrm{Supp}(Y)} \mathcal{X}_y\right| < 2^\ell \cdot 2^{k-\ell-t} = 2^{k-t}.$$

It follows that

$$\mathbb{P}_{(x,y)\overset{\mathrm{R}}{\leftarrow}(X,Y)}\left[\underset{X|Y}{\mathrm{H}}(x|y) < k - \ell - t\right] \le \mathbb{P}_{(x,y)\overset{\mathrm{R}}{\leftarrow}(X,Y)}[x \in \mathcal{X}] < 2^{-k} \cdot 2^{k-t} = 2^{-t}.$$

□

Smoothed entropies. The following lemma implies that a random variable X whose sample-entropy is high, with high probability, behaves like a random variable that has high min-entropy (i.e., with sample-entropy function that is "smoother" without any peaks).

Lemma C.4. *Let X and Y be random variables and let $\varepsilon > 0$.*

i. *Suppose* $\mathbb{P}_{x \xleftarrow{R} X}[\mathrm{H}_X(x) \geq k] \geq 1-\varepsilon$. *Then* X *is* ε*-close to a random variable* X' *with* $\mathrm{H}_\infty(X') \geq k$.

ii. *Suppose* $\mathbb{P}_{(x,y) \xleftarrow{R} (X,Y)}\left[\mathrm{H}_{X|Y}(x|y) \geq k\right] \geq 1-\varepsilon$. *Then* (X,Y) *is* ε*-close to a random variable* (X',Y') *with* $\mathrm{H}_{X'|Y'}(x|y) \geq k$ *for any* $(x,y) \in \mathrm{Supp}(X',Y')$. *Further,* Y' *and* Y *are identically distributed.*

Proof. For the first item, one can modify X on an ε fraction of the probability space (corresponding to when X takes on a value x such that $\mathrm{H}_X(x) \geq k$) to bring all probabilities to be smaller than or equal to 2^{-k}.

The second item is proved via similar means, while when changing (X,Y), we do so without changing the "Y" coordinate. □

Flattening Shannon entropy. It is known that, up to small statistical distance, the Shannon entropy of a random variable can be converted to min-entropy by taking independent copies of this variable. We state the more exact version from [Yan15]:

Lemma C.5 ([Yan15], Theorem 3.14)**.** *Let* X *be a random variable taking values in a universe* $\mathcal{U}$*, let* $t \in \mathbb{N}$*, and let* $0 < \varepsilon \leq 1/e^2$*. Then with probability at least* $1-\varepsilon$ *over* $x \xleftarrow{R} X^{(t)}$*,*

$$\underset{X^{(t)}}{\mathrm{H}}(x) \geq t \cdot \mathrm{H}(X) - \sqrt{8t \cdot \log \tfrac{1}{\varepsilon}} \cdot \left(\log |\mathcal{U}| + \frac{1}{2}\log t\right).$$

We will make use of the following "conditional variant" of Lemma C.5:

Lemma C.6. *Let* X *and* Y *be jointly distributed random variables where* X *takes values in a universe* $\mathcal{U}$*, let* $t \in \mathbb{N}$*, and let* $0 < \varepsilon \leq 1/e^2$*. Then with probability at least* $1-\varepsilon$ *over* $(x,y) \leftarrow (X',Y') = (X,Y)^{(t)}$*,*

$$\underset{X'|Y'}{\mathrm{H}}(x \mid y) \geq t \cdot \mathrm{H}(X \mid Y) - \sqrt{8t \cdot \log \tfrac{1}{\varepsilon}} \cdot \left(\log |\mathcal{U}| + \frac{1}{2}\log t\right).$$

Proof. Follows the same line as the proof of Lemma C.5, by considering the random variable $\mathrm{H}_{X|Y}(X|Y)$ instead of $\mathrm{H}_X(X)$. □

Sequence of random variables. For measuring the real and accessible entropy, we can measure the entropy of a subset of random variables, conditioned on the previous elements in the sequence. The following lemma generalizes Lemma C.2 to some other settings that come up when measuring the accessible entropy.

Definition C.7. *For a* t*-tuple random variable* $\mathbf{X} = (X_1, \ldots, X_t)$*,* $\mathbf{x} \in \mathrm{Supp}(X)$ *and* $\mathcal{J} \subseteq [t]$*, let*

$$\underset{\mathbf{X},\mathcal{J}}{\mathrm{H}}(\mathbf{x}) = \sum_{i \in \mathcal{J}} \underset{\mathbf{X}_i|\mathbf{X}_{<i}}{\mathrm{H}}(\mathbf{x}_i|\mathbf{x}_{<i}).$$

The following fact is immediate by the chain rule.

Proposition C.8. *Let* $\mathbf{X} = (X_1, \ldots, X_t)$ *be a sequence of random variables and let* $\mathcal{J} \subseteq [t]$*. Then,* $\mathrm{E}_{\mathbf{x} \xleftarrow{R} \mathbf{X}}[\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x})] = \sum_{j \in \mathcal{J}} \mathrm{H}(X_j|\mathbf{X}_{<j}) \leq \mathrm{H}_0(\mathbf{X}_{\mathcal{J}})$.

Proof. Compute

$$\begin{aligned}
\mathrm{E}_{\mathbf{x}\xleftarrow{R}\mathbf{X}}\left[\underset{\mathbf{X},\mathcal{J}}{\mathrm{H}}(\mathbf{x})\right] &= \mathrm{E}_{\mathbf{x}\xleftarrow{R}\mathbf{X}}\left[\sum_{j\in\mathcal{J}} \underset{X_j|\mathbf{X}_{<j}}{\mathrm{H}}(\mathbf{x}_j|\mathbf{x}_{<j})\right] \\
&= \sum_{j\in\mathcal{J}} \mathrm{E}_{\mathbf{x}\xleftarrow{R}\mathbf{X}}\left[\underset{X_j|\mathbf{X}_{<j}}{\mathrm{H}}(\mathbf{x}_j|\mathbf{x}_{<j})\right] \\
&= \sum_{j\in\mathcal{J}} \mathrm{H}(X_j|\mathbf{X}_{<j}) \\
&\le \sum_{j\in\mathcal{J}} \mathrm{H}_0(X_j) \le \mathrm{H}_0(\mathbf{X}_{\mathcal{J}}).
\end{aligned}$$

□

The following lemma generalizes Lemma C.2 to such a sequence of random variables.

Lemma C.9. *Let $\mathbf{X} = (X_1, \dots, X_t)$ be a sequence of random variables and let $\mathcal{J} \subseteq [t]$. If $\mathbb{P}_{\mathbf{x}\xleftarrow{R}\mathbf{X}}[\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x}) > k] \le \varepsilon \in [0,1]$, then $\mathrm{E}_{\mathbf{x}\xleftarrow{R}\mathbf{X}}[\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x})] \le (1-\varepsilon)k + \varepsilon \cdot (\mathrm{H}_0(\mathbf{X}_{\mathcal{J}}) - \log 1/\varepsilon) \le k + \varepsilon \cdot \mathrm{H}_0(\mathbf{X}_{\mathcal{J}}) + 1$.*

Proof. The proof is similar to that of Lemma C.2. Let $\mathcal{Y} = \{\mathbf{x} \in \mathrm{Supp}(\mathbf{X})\colon \mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x}) > k\}$, let $\mathbf{Y} = \mathbf{X}|_{\mathbf{X}\in\mathcal{Y}}$, and for $\mathbf{y}_{\le k} \in \mathrm{Supp}(\mathbf{Y}_{<j})$ let $\varepsilon_{\mathbf{y}_{\le j}} = \frac{\mathbb{P}[\mathbf{Y}_j=\mathbf{y}_j|\mathbf{Y}_{<j}=\mathbf{y}_{<j}]}{\mathbb{P}[\mathbf{X}_j=\mathbf{y}_j|\mathbf{X}_{<j}=\mathbf{y}_{<j}]}$. Compute

$$\begin{aligned}
\mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\underset{\mathbf{Y},\mathcal{J}}{\mathrm{H}}(\mathbf{y})\right] &= \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in\mathcal{J}} \underset{\mathbf{Y}_i|\mathbf{Y}_{<j}}{\mathrm{H}}(\mathbf{y}_j|\mathbf{y}_{<j})\right] \\
&= \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in\mathcal{J}} \underset{\mathbf{X}_j|\mathbf{X}_{<j}}{\mathrm{H}}(\mathbf{y}_j|\mathbf{y}_{<j}) + \log\frac{\mathbb{P}[\mathbf{Y}_j=\mathbf{y}_j|\mathbf{Y}_{<j}=\mathbf{y}_{<j}]}{\mathbb{P}[\mathbf{X}_j=\mathbf{y}_j|\mathbf{X}_{<j}=\mathbf{y}_{\le j}]}\right] \\
&= \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in\mathcal{J}} \underset{\mathbf{X}_j|\mathbf{X}_{<j}}{\mathrm{H}}(\mathbf{y}_j|\mathbf{y}_{<j}) + \log\varepsilon_{\mathbf{y}_{\le j}}\right] \\
&= \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in\mathcal{J}} \underset{\mathbf{X}_j|\mathbf{X}_{<j}}{\mathrm{H}}(\mathbf{y}_j|\mathbf{y}_{<j})\right] + \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in\mathcal{J}} \log\varepsilon_{\mathbf{y}_{\le j}}\right], \qquad (24)
\end{aligned}$$

and note that

$$\begin{aligned}
\mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in\mathcal{J}} \log\varepsilon_{\mathbf{y}_{\le j}}\right] &\ge \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\sum_{j\in[t]} \log\varepsilon_{\mathbf{y}_{\le j}}\right] \qquad (25) \\
&= \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\log\prod_j \varepsilon_{\mathbf{y}_{\le j}}\right] \\
&= \mathrm{E}_{\mathbf{y}\xleftarrow{R}\mathbf{Y}}\left[\log\frac{\mathbb{P}[\mathbf{Y}=\mathbf{y}]}{\mathbb{P}[\mathbf{X}=\mathbf{y}]}\right] \\
&= \log 1/\mathbb{P}[\mathbf{X}\in\mathcal{Y}].
\end{aligned}$$

Let $\varepsilon' = \mathbb{P}[\mathbf{X} \in \mathcal{Y}]$. We conclude that

$$\begin{aligned}
\mathrm{E}_{\mathbf{x} \xleftarrow{R} \mathbf{X}}\left[\underset{\mathbf{Y},\mathcal{J}}{\mathrm{H}}(\mathbf{x})\right] &= \sum_{\mathbf{x} \in \mathrm{Supp}(\mathbf{X}) \setminus \mathcal{Y}} \mathbb{P}[\mathbf{X} = \mathbf{x}] \cdot \underset{\mathbf{X},\mathcal{J}}{\mathrm{H}}(\mathbf{x}) + \sum_{\mathbf{x} \in \mathcal{Y}} \mathbb{P}[\mathbf{X} = \mathbf{x}] \cdot \underset{\mathbf{X},\mathcal{J}}{\mathrm{H}}(\mathbf{x}) \\
&\leq (1 - \varepsilon')k + \varepsilon' \cdot \sum_{\mathbf{x} \in \mathcal{Y}} \mathbb{P}[\mathbf{Y} = \mathbf{x}] \cdot \underset{\mathbf{X},\mathcal{J}}{\mathrm{H}}(\mathbf{x}) \\
&\leq (1 - \varepsilon')k + \varepsilon' \cdot (\mathrm{E}_{\mathbf{y} \xleftarrow{R} \mathbf{Y}}\left[\underset{\mathbf{Y},\mathcal{J}}{\mathrm{H}}(\mathbf{y})\right] - \log 1/\varepsilon') \\
&\leq (1 - \varepsilon')k + \varepsilon' \cdot (\mathrm{H}_0(\mathbf{X}_{\mathcal{J}}) - \log 1/\varepsilon') \\
&\leq (1 - \varepsilon)k + \varepsilon \cdot (\mathrm{H}_0(\mathbf{X}_{\mathcal{J}}) - \log 1/\varepsilon) \\
&\leq k + \varepsilon \cdot \mathrm{H}_0(\mathbf{X}_{\mathcal{J}}) + 1.
\end{aligned}$$

□

The next two lemmas are only needed when measuring the *max* accessible entropy.

Lemma C.10. *Let $\mathbf{X} = (X_1, \ldots, X_t)$ be a sequence of random variables and let $\mathcal{J} \subseteq [t]$. Then,*

i. $\mathrm{E}_{\mathbf{x} \xleftarrow{R} \mathbf{X}}\left[2^{\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x})}\right] \leq \mathrm{H}_0(\mathbf{X}_{\mathcal{J}})$.

ii. $\mathbb{P}_{x \xleftarrow{R} X}\left[\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x}) > \log \frac{1}{\varepsilon} + \mathrm{H}_0(X_{\mathcal{J}})\right] < \varepsilon$, *for any* $\varepsilon > 0$.

Proof. The second item follows from the first one as in the proof of Lemma C.1. We prove the first item by induction on t and $|J|$. The case $t = 1$ is immediate, so we assume for all $(t', \mathcal{J}')$ with $(t', |\mathcal{J}'|) < (t, |\mathcal{J}|)$ and prove it for $(t, \mathcal{J})$. Assume that $1 \in \mathcal{J}$ (the case $1 \notin \mathcal{J}$ is analogous) and let $\mathbf{X}_{-1} = (X_2, \ldots, X_t)$ and $\mathcal{J}_{-1} = \{i - 1 \colon i \in \mathcal{J} \setminus \{1\}\}$. Compute

$$\begin{aligned}
\mathrm{E}_{\mathbf{x} \xleftarrow{R} \mathbf{X}}\left[2^{\mathrm{H}_{\mathbf{X},\mathcal{J}}(\mathbf{x})}\right] &= \sum_{x_1 \in \mathrm{Supp}(X_1)} 2^{-\mathrm{H}_{X_1}(\mathbf{x}_1)} \cdot 2^{\mathrm{H}_{X_1}(x_1)} \cdot \mathrm{E}_{\mathbf{x} \xleftarrow{R} \mathbf{X}_{-1}|_{X_1 = x_1}}\left[2^{\mathrm{H}_{\mathbf{X}_{-1}|_{X_1=x_1},\mathcal{J}_{-1}}(\mathbf{x})}\right] \\
&\leq \sum_{x_1 \in \mathrm{Supp}(X_1)} 1 \cdot \left|\mathrm{Supp}((\mathbf{X}_{-1})_{\mathcal{J}_{-1}}|_{X_1=x_1})\right| \\
&= \sum_{x_1 \in \mathrm{Supp}(X_1)} \left|\mathrm{Supp}(\mathbf{X}_{\mathcal{J}\setminus\{1\}}|_{X_1=x_1})\right| \\
&= |\mathrm{Supp}(\mathbf{X}_{\mathcal{J}})|.
\end{aligned}$$

□

Sub-additivity. The chain rule for Shannon entropy yields that

$$\mathrm{H}(X = (X_1, \ldots, X_t)) = \sum_i \mathrm{H}(X_i | X_1, \ldots, X_{i-1}) \tag{26}$$

$$\leq \sum_i \mathrm{H}(X_i). \tag{27}$$

The following lemma shows that a variant of the above also holds for sample-entropy.

Lemma C.11. *For random variables* $\mathbf{X} = (X_1, \ldots, X_t)$*, it holds that*

i. $\mathrm{E}_{\mathbf{x} \xleftarrow{R} \mathbf{X}}\left[2^{\mathrm{H}\mathbf{x}(\mathbf{x}) - \sum_t \mathrm{H}_{X_i}(\mathbf{x}_i)}\right] \leq 1$*, and*

ii. $\mathbb{P}_{\mathbf{x} \xleftarrow{R} \mathbf{X}}\left[H_{\mathbf{X}}(\mathbf{x}) > \log \frac{1}{\varepsilon} + \sum_{i \in [t]} H_{X_i}(\mathbf{x}_i)\right] < \varepsilon$*, for any* $\varepsilon > 0$*.*

Proof. As in Lemma C.1, the second part follows from the first by Markov's inequality. For the first part, compute

$$\begin{aligned}
\mathrm{E}_{\mathbf{x} \xleftarrow{\mathrm{R}} \mathbf{X}}\left[2^{\mathrm{H}\mathbf{x}(\mathbf{x}) - \sum_t \mathrm{H}_{X_i}(\mathbf{x}_i)}\right] &= \sum_{\mathbf{x} \in \mathrm{Supp}(\mathbf{X})} \mathbb{P}[\mathbf{X} = \mathbf{x}] \cdot \frac{\prod_{i \in [t]} \mathbb{P}[X_i = \mathbf{x}_i]}{\mathbb{P}[\mathbf{X} = \mathbf{x}]} \\
&= \sum_{\mathbf{x} \in \mathrm{Supp}(\mathbf{X})} \prod_i \mathbb{P}[X_i = \mathbf{x}_i] \\
&\leq 1.
\end{aligned}$$

□