中国科学院大学

University of Chinese Academy of Sciences

代码水印综述

作者姓名：赵睿智

学位类别：工学硕士

学科专业：信息网络安全

培养单位：中国科学院网络空间安全学院

2025 年 5 月

Overview of Code Watermarks

A thesis submitted to

Literature reading course Of

Cyberspace security evaluation In

University of Chinese Academy of Sciences

By

Zhao Ruizhi

Computer Science

Information Network Security

School of Cyberspace Security, Chinese Academy of Sciences

May, 2025

摘 要

随着代码生成技术与大语言模型的广泛应用，源代码的版权保护与归属可追溯性问题日益受到关注。作为一种主动防护手段，源码水印技术通过在不影响代码语义和可编译性的前提下，将隐蔽的标识信息嵌入源代码中，提供可验证的所有权标识。近年来，研究者围绕水印嵌入方式、提取机制、鲁棒性设计与对抗攻击防护等方面提出了多种方法，并在多语言、多任务场景中展开应用探索。本文系统梳理了源码水印的研究现状，归纳了主流方法的技术路线和分类体系，深入分析了当前面临的关键挑战，包括低熵特性下的嵌入空间受限、对抗语义变换的鲁棒性不足、水印可用性与可扩展性缺失、以及伦理和隐私风险等问题。最后，本文展望了未来研究方向，强调将源码水印技术与程序语义建模、形式化验证、安全机制以及标准化协议相结合，以实现更高效、可信且可部署的水印系统。

关键词：源码水印；代码归属；鲁棒性；语义变换；大语言模型；对抗攻击；水印检测；隐私保护

Abstract

With the rise of neural code generation and large language models, the protection and traceability of source code ownership have become increasingly critical. Source code watermarking, as a proactive defense technique, embeds imperceptible ownership information into code without altering its semantics or functionality, enabling verifiable attribution and misuse prevention. In recent years, extensive efforts have been made to design watermark embedding schemes, extraction mechanisms, and robustness strategies, along with applications across multi-language and multi-task settings. This paper presents a comprehensive survey of source code watermarking, categorizing existing approaches based on technical frameworks and abstraction levels, and analyzing key challenges faced by the field—such as limited embedding capacity due to low entropy of code, vulnerability to semantic-preserving transformations, lack of usability and scalability, and emerging concerns regarding ethics and privacy. Finally, the paper outlines promising future directions, including combining watermarking with semantic program analysis, formal verification, cryptographic guarantees, and standardization efforts, aiming to build more robust, efficient, and deployable watermarking systems for real-world applications.

Keywords: source code watermarking; code attribution; robustness; semantic-preserving transformation; large language models; adversarial attacks; watermark detection; privacy-preserving verification

目 录

第 1 章　引言

1.1　研究背景

随着软件产业的持续扩张与开源社区的快速发展，源代码的复制、篡改与非法传播问题日益严重，特别是在人工智能辅助编程（如 GitHub Copilot、ChatGPT 等）日益普及的背景下，源代码的归属判定与可追溯性问题变得更加突出。一方面，未经授权的代码再分发可能侵犯原作者版权；另一方面，AI 生成代码可能“继承”训练数据中的开源片段，造成潜在的知识产权风险。此外，在软件安全、学术诚信与数字取证等场景中，确立源代码的所有权和来源已成为一项迫切需求。

为应对上述挑战，源码水印（Source Code Watermarking）技术应运而生。该技术通过在不影响代码功能与语义的前提下，向源代码中嵌入隐蔽的标识信息（如作者身份、版权声明、版本信息等），从而实现代码归属的主动声明与后期可验证追踪。相比传统的代码克隆检测或作者风格分析，源码水印具备更强的主动性、显式性与抗篡改能力，为代码追溯提供了技术基础。

尽管数字水印在图像、音频、自然语言文本等领域已有广泛应用，但将其迁移到源代码场景中仍面临独特挑战：编程语言的语法与语义约束极为严格，稍有不当修改就可能破坏代码可编译性或功能正确性；同时，代码具备“自然通道”（面向人类的可读性）与“形式通道”（面向编译器的语义执行）双重属性，给水印设计带来更高复杂性。此外，面对变量重命名、语义重构、格式扰动等攻击手段，以及重水印（rewatermarking）引发的归属冲突问题，现有方案在鲁棒性与可信性上仍有明显不足。

近年来，研究者提出了一系列针对源码水印的解决方案，涵盖语义保持的代码变换、AST 级水印嵌入、双通道编码模型、深度学习辅助嵌入提取等策略。本文旨在对这一领域的最新研究进行系统综述，梳理技术演进路径，总结现有方法的设计思路、优势与局限，分析当前面临的关键挑战，并探讨未来的发展方向。

1.2　主要贡献

本综述的主要贡献如下：

1. 明确源码水印的研究背景与核心问题，构建问题定义框架；
2. 从方法视角对现有技术进行分类与对比分析；
3. 系统讨论水印系统在嵌入策略、提取机制、鲁棒性设计等方面的进展；
4. 识别当前研究空白与挑战，提出未来可能的发展趋势与研究建议。

第 2 章　核心问题

源码水印（Source Code Watermarking）技术，作为一种主动防御机制，致力于将可识别的所有权标识（如作者 ID、时间戳、版权声明等）以隐蔽、鲁棒的方式嵌入源代码中，并可在后期通过算法手段进行提取和验证，以用于版权声明、伪冒检测或源代码追责。

源码水印的核心研究问题可归纳为以下几个方面：

1. 水印嵌入 (Watermark Embedding)：

如何在保持代码语法正确、行为一致的前提下，将水印信息（如身份 ID、时间戳、追踪标识等）嵌入源代码中？这通常依赖语义保持的代码变换、抽象语法树（AST）操作或变量命名控制等手段。

2. 水印提取 (Watermark Extraction)：

如何在代码被部分修改或混淆的情况下，准确地从目标代码中还原出原始水印信息？系统需支持高效、鲁棒的解码机制，并避免误提取与伪阳性。

3. 鲁棒性 (Robustness)：

水印应能抵抗常见的攻击行为，如变量重命名、语义保持的重构、格式扰动等。系统应在对抗攻击下仍保留水印有效性。

4. 透明性 (Transparency)：

水印嵌入应对原始代码尽可能不可察觉，不应降低其可读性、风格一致性或开发维护体验，尤其在开源场景中尤为重要。

5. 容量与唯一性 (Capacity & Uniqueness)：

系统应能嵌入足够长度的水印比特串，以支持大规模唯一标识。同时，应评估不同水印之间的可区分性，避免冲突或重复。

6. 重水印攻击与归属歧义评估 (Re-watermarking & Ownership Ambiguity Assessment)：

在现实中，攻击者可能通过嵌入新的水印覆盖原始水印，以伪造代码归属。这引出了新的问题：

(a) 归属判别问题：如何识别哪个水印是原始嵌入？如何防止归属争议中的“先后不明”？

(b) 覆盖鲁棒性评估：系统应对重水印具备抗覆盖能力，确保原水印不会被完全替代；

(c) 可信提取机制设计：是否可引入数字签名、时间戳、加密编码等手段提升水印来源的不可否认性与可信度?

第 3 章　研究方法与技术分类

3.1　前置思路

保护大模型生成的源代码要求通过生成的代码来跟踪其来源，以便在出现争议时作者声称其所有权。克隆检测 (Roy 等, 2007; Koschke, 2007; Bellon 等, 2007) 和作者归属是最先提出的针对该问题的解决方案。克隆检测是一种用于识别和检测代码中重复或相似片段（称为“克隆”）的方法，迄今为止已出现基于文本的、基于令牌的、基于树的、基于 pdg 的、基于指标的和混合的代码克隆检测算法 (Büch 等, 2019); 针对源码的作者归属技术 (Burrows 等, 2014) 起源于对自然语言的作者的身份鉴别，其通过将程序员与基于程序员独特的风格特征的给定代码相关联来识别代码作者的过程。然而，这一问题既困难又不同于自然语言文本的作者身份识别，困难主要来源于编译器的语法规则所建立的编写的代码表达式固有的不灵活性。上述方案均依赖于代码生成后本身隐式和被动的特征 (比如代码风格、功能等)，无法主动改变其特征以用来添加相应的身份标识。

3.2　基于内容的文本水印嵌入

在此基础之上，出现了将数字水印嵌入源码并提取的思路。这是一种主动为代码施加身份标识 (即水印) 并从中提取并验证或追溯其来源的手段。该思路来源于对大模型生成内容的水印施加和检测，随着诸如 ChatGPT 等大模型系统变得越来越普遍，它们可能被用于恶意目的的风险也越来越大 (Bergman 等, 2022; Mirsky 等, 2023) 。其中包括利用社交媒体平台上的自动机器人进行社会工程和选举操纵活动，制造假新闻和网络内容，以及利用人工智能系统在学术写作和编码作业中作弊。而大模型生成的代码本身便属于针对人类提问回答的一部分，因此研究者提出了将针对大模型生成文本的水印施加、检测以及提取的方法应用于大模型生成的代码中。整体而言，这类水印施加方法基于单词替换 (Topkara 等, 2006b) 或语句释义 (Topkara 等, 2006a) 的方法，按照水印嵌入的时机以及嵌入方式的区别又可分为硬水印和软水印两种类型。硬水印编码使用 BERT (Devlin 等, 2019)、RoBERTa (Liu 等, 2019) 这样的掩码语言模型，将生成内容中的词替换为同义词。这种方法在不同输入下插入的水印模式相似，因此容易被识别和

去除，安全性较低；软水印嵌入即在模型生成过程中动态调整词汇的采样概率，从而隐式嵌入水印。这样的方式可以根据输出内容变化，水印嵌入模式更隐蔽，安全性更强，因此也成为水印研究的主流方向。WLLM(用于大语言模型的水印) (Kirchenbauer 等, 2023) 是一种通过红绿表为大模型施加软水印的实施方案，该方法在每个生成步骤中随机地将整个词表划分为两个组（即绿色列表和红色列表），并提高绿色列表中词元被采样的概率。具体做法是：对绿色列表中的词元的 logits 加上一个标量值，从而使模型在生成时更倾向于选择绿色列表中的词元，而不是红色列表的。虽然这种基于水印的方法和即时检测方法在许多语言生成任务中都能很好地工作，但性能不能很好地转移到代码生成任务中。这是因为代码是基于相较于自然语言更受限的语法规则生成，而且需要严格保证其可执行性与输出结果的正确性，导致其熵值较低。一方面，若用此方式为代码施加强度较高的水印，会严重降低代码的质量与可用性；若施加水印强度过弱，则代码本身的低熵性会导致水印嵌入的选择范围过小，难以嵌入的同时增加了水印检测的难度。在此基础上，Taehyun L 等研究者拓展了 WLLM 并提出了 SWEET(Selective WatErmarking via Entropy Thresholding) 方法 (Lee 等, 2023)，适用于代码大语言模型（Code LLMs）以及一般的大语言模型（LLMs）。与对每个生成词元都应用“绿色-红色”规则不同，SWEET 仅对熵高于某个阈值的词元应用该规则。即对于在生成功能性代码中起关键作用的词元不应用绿色-红色规则；而对不那么重要的部分确保嵌入足够多的绿色列表词元，以构造一个可检测的水印。这种选择性嵌入水印的策略可从一定程度上缓解语义替换与代码语法规则的冲突，但其仍存在以下问题：只嵌入一比特的水印，只能表明该代码是否存在水印，不足以保留足够的来源信息 (如 LLM 的供应商 ID 等)；水印嵌入模式较为单一且通道较为有限，容易受到替换或是混淆等攻击的影响。

传统的文本水印方法难以直接应用于代码领域，本质在于编程语言对语法和语义的严格性使得水印嵌入空间有限。迁移大模型生成内容的水印嵌入方式到生成的代码中均不可避免地修改了对程序逻辑至关重要的标记，例如条件表达式中的关键字或算术计算中的操作符。这些修改可能会引入语法错误或改变代码的行为，从而破坏其可靠性。因此需要对文本水印嵌入模式进行代码语法规则约束方向的优化改良。BOQUAN LI 等提出了一种新颖的水印方法 ACW (Li 等, 2024)，用于追踪和识别由大语言模型（如 ChatGPT、StarCoder 等）生成的代

码。ACW 设计了 46 种语义保持且幂等的代码变换规则，这些规则分为三类：重构（如去除多余的 else 分支）、重排序（如根据哈希值调整操作数顺序）和格式调整（如空格、缩进统一）。在水印嵌入阶段，ACW 根据特定的哈希顺序选择性地应用这些变换，利用其“是否已应用”来编码水印信息。同时 ACW 还引入了 BCH 纠错码来增强水印提取的鲁棒性，即使部分水印信息被破坏，也能恢复原始水印。完全基于规则的 ACW 的独特之处在于它完全基于预定义的规则，无需训练模型，并且在一定程度上对代码优化工具有抵抗性；而由 Jungin Kim 等研究者提出的 STONE(Syntax TOkeN preserving codE watermark) (Kim 等, 2025) 则通过其独特的语法感知能力，仅在非语法关键的 token 上嵌入水印，避免对关键字、运算符等语法元素的干扰，从而保障代码的正确性与可执行性。为评估水印的有效性，论文还引入了一个新的综合评估指标 CWEM，涵盖了代码的正确性（Correctness）、可检测性（Detectability）和自然性（Naturalness）。实验结果表明，STONE 相比现有方法（如 SWEET、EWD 等）在 Python、C++ 和 Java 上具有更好的水印效果和更小的功能损害，同时具有更低的检测开销，展示了其在实际代码生成场景中的实用性与鲁棒性。来自华中科技大学的研究者们提出了 CODEIP 这一多位软水印嵌入方法 (Guan 等, 2024) ，它基于 LLM 代码生成的语法引导，在代码生成过程中根据 LLM 的概率逻辑插入水印消息，从而在生成的代码中嵌入多比特的信息。此外，CODEIP 将语法信息集成到生成带水印代码的过程中，通过训练类型预测器来预测下一个标记的后续语法类型来最大限度地提高生成语义正确代码的可能性。

3.3　基于双通道理论的源码水印嵌入

基于文本水印对于源码本身语法规则的随意性考量，Casey C 等研究者提出了源码的双通道理论 (Casalnuovo 等, 2020) 。源代码相较于自然语言之区别在于其包含两种信息渠道，分别是指定计算机执行的算法（AL）通道和解释该执行的目的和上下文的自然语言（NL）通道。其中 AL 通道从代码的语义派生其含义，其涵盖了代码中的控制逻辑与数据流逻辑。该通道既针对编译并执行的计算机程序，也面向于需维护并理解其含义的工作者；NL 通道涵盖数据流中变量的标识符名称以及代码注释内容，他们不会影响程序的执行过程，但会影响人类对程序逻辑语义的理解过程。如：变量名的命名内容以及风格会对程序开发者理

解该段代码的功能产生巨大影响。从这一层面上也可引出代码水印这一技术与代码混淆技术的区别 (Balakrishnan 等, 2005)：代码混淆技术通过变量命名随机化、字符串加密等方式以最大程度上干扰开发者对其程序语义的理解，从而达到保护代码以及知识产权的目的。因此，从源代码上嵌入数字水印需要兼顾上述两个通道的限制。对于 AL 通道而言，水印的施加需要保证代码执行逻辑与结果的不变性，同时保证其具有与施加水印前代码相当的执行效率；对于 NL 通道而言，水印的施加应尽可能保持代码语义上的可读性，同时尽可能减少理解成本 (Casalnuovo 等, 2020; Yang 等, 2023) 。LLM 生成的源码是公开透明的，因此源码水印的施加需要充分考虑到其隐蔽性。

对于 NL 通道而言，该部分的内容本身就是直接与阅读者进行交互的内容，同时不受机器执行规则的束缚，因而灵活性较大。但也正是因为此原因，导致这部分内容同时与源码本身相关性较弱，易于被攻击者察觉并去除；对于 AL 通道而言，该部分内容是源码逻辑的核心部分，同时也是知识版权保护的重点模块。但限于代码语法规则的完备性，对运行设备而言其必须是明确可执行的高级指令，因此基于此的变换形式相当有限，限制了施加水印的信息量。因此需要平衡两个通道的优势与不足之处，以完善代码水印的鲁棒性。SrcMarker (Yang 等, 2023) 是一种基于双通道理论，结合规则嵌入模式与自学习框架的新型源码水印嵌入系统。该系统聚焦于函数级别的源代码片段，提出了基于“双通道水印嵌入策略”的设计理念，分别从代码的自然通道（如变量命名）和形式通道（如控制结构、语句变换）两个角度对代码进行语义保持的改写，从而将水印比特串以可控、隐蔽的方式编码进源代码中。

为了实现跨语言、结构一致的代码变换，SrcMarker 构建了一个语言无关的抽象语法树表示框架 MutableAST，允许研究者以统一方式对 C、C++、Java、JavaScript 等语言的代码进行结构化变换，并保障变换的语义等价性。此外，为解决代码变换的不可微特性与深度模型训练的不兼容问题，SrcMarker 提出了一种特征空间近似机制（Feature-Space Approximation），将离散的代码操作投影到连续的特征空间中，从而实现水印嵌入与提取模块的端到端可微训练。

系统整体由三部分组成：嵌入模块通过神经网络根据输入的水印比特与代码上下文选择变换操作；提取模块则利用编码后的代码特征还原水印；近似模块则桥接不可微变换与可训练网络之间的梯度路径。此外，SrcMarker 在面向函数

和项目两级的真实代码数据集上进行了全面评估，实验表明其在水印嵌入准确率、鲁棒性（对抗代码扰动和重水印攻击）、执行正确性和自然性保留方面均显著优于现有方法。

3.4　挑战与未来方向

尽管源码水印技术在近年来取得了显著进展，但在理论研究和实际应用中仍面临多方面的挑战。首先，源代码自身的特性构成了水印嵌入的天然障碍。代码具有较低的熵和高度结构化的语法，这使得可用于嵌入信息的冗余空间有限，水印既难嵌入，又容易被检测。此外，代码必须严格遵循语法和语义规则，稍有不当就可能导致编译失败或功能错误，因此嵌入操作需极度精细，且必须保持代码功能完全等价。为克服这些限制，未来研究可探索基于深层语义表示的水印编码方法，引入形式化验证机制保障语义一致性，或挖掘对程序行为影响最小的“自由度”进行隐蔽编码。

当前基于源码保护的水印系统正面临日益复杂的攻击手段。包括语义保持重写、代码混淆、基于大语言模型（LLM）的释义改写、自适应去水印攻击，以及多个副本对比下的串通攻击等。这些攻击通常能在不破坏程序功能的前提下，悄然移除水印。DeCoMa (Xiao 等, 2025) 提出了一个首个专门面向代码数据集水印检测与清除的攻击框架。该系统基于对代码的双通道抽象映射（Dual-Channel Abstraction），将源代码的自然通道（如变量名、注释）与形式通道（如控制结构、表达式）进行统一模板化表示，从中识别出可疑的“触发-目标”水印模式对，并以异常对检测方法（如 z-score 离群识别）提取嵌入式水印。随后，DeCoMa 能有效清除这些水印样本，实现对原始受保护数据集的“净化”，从而绕过水印验证。实验覆盖 14 种水印场景，结果显示 DeCoMa 在检测准确率、效率与模型保真性方面均显著优于现有基线，且无需模型训练，速度快至百倍以上。该工作首次系统展示了代码水印可被自动识别与去除的可行性，并对未来水印机制的对抗性提出了重大挑战。由此可知，未来系统需构建与核心逻辑紧密耦合、难以被替代的水印机制；开发能抗击 LLM 重写的嵌入策略；并结合密码学技术设计安全、抗串通的指纹编码方法。同时，还可借助程序不变量或抽象语义特征增强水印的稳健性。

要实现源码水印技术的实际落地，还需解决可用性、扩展性和标准化等方面

的问题。目前的水印系统在嵌入或验证时往往依赖暴露水印信息，流程繁琐、缺乏隐私保护；在大规模系统中难以支持数十万级别的唯一标识编码；缺乏统一的评测基准也阻碍了不同方案间的公平比较。此外，水印处理的效率对于工业部署而言仍显不足。未来研究可致力于开发轻量级、可扩展的水印算法，制定统一的检测协议与元数据格式，引入硬件加速和零知识验证等机制提升实用性和隐私性。

最后，源码水印的推广必须充分考虑伦理与社会影响。水印机制若被滥用，可能被用于虚假归属或代码审查操控，引发版权纠纷甚至隐私侵犯。同时，在开源社区中，水印技术可能与自由传播和协作开发的精神产生冲突。此外，其技术原理也可能被滥用于恶意目的，如信息隐蔽通道或增强恶意代码隐匿性。因此，未来需同步推进水印系统的伦理治理，制定公平透明的开发规范，确保技术的责任使用，在保护知识产权的同时不损害用户权益与社会信任。

综上所述，源码水印的持续发展依赖于跨层次的协同进步。仅有算法性能的提升尚不足够，系统必须在安全性、可用性、可扩展性与伦理合法性之间取得平衡，才能真正实现广泛应用与可持续创新。

3.5 总结

源码水印作为保障代码归属与追溯的重要手段，已逐步成为 AI 时代背景下软件知识产权保护、学术诚信监管与生成内容标识的关键技术之一。本文系统回顾了该领域的发展脉络，从水印嵌入与提取机制、鲁棒性设计、功能保持策略到代表性方法的技术路径与实验表现，全面梳理了当前研究的主要方向与方法体系。同时，我们深入分析了该领域所面临的核心挑战，包括代码自身的结构限制、复杂对抗攻击、实用部署的障碍以及伦理风险等。

展望未来，源码水印的研究亟需在提升嵌入容量与鲁棒性的同时，兼顾系统的可扩展性、隐私保护能力与跨语言适应性。技术创新应与标准化、可用性设计和伦理规范建设同步推进，构建可落地、可验证、可信赖的完整解决方案。此外，随着大语言模型的普及和 AI 生成代码的广泛部署，源码水印也将在 AI 安全和责任追踪等新兴应用中发挥更为核心的作用。

源码水印作为程序语言处理、AI 生成内容治理与数字版权保护的重要交叉点，将在未来的研究与实践中持续扮演关键角色，值得持续投入与深入探索。

参考文献

Balakrishnan A, Schulze C. Code obfuscation literature survey [J]. CS701 Construction of compilers, 2005, 19(31): 10.

Bellon S, Koschke R, Antoniol G, et al. Comparison and evaluation of clone detection tools [J]. IEEE Transactions on software engineering, 2007, 33(9): 577-591.

Bergman A S, Abercrombie G, Spruit S, et al. Guiding the release of safer e2e conversational ai through value sensitive design [C]//Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue. Association for Computational Linguistics, 2022.

Büch L, Andrzejak A. Learning-based recursive aggregation of abstract syntax trees for code clone detection [C]//2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2019: 95-104.

Burrows S, Uitdenbogerd A L, Turpin A. Comparing techniques for authorship attribution of source code [J]. Software: Practice and Experience, 2014, 44(1): 1-32.

Casalnuovo C, Barr E T, Dash S K, et al. A theory of dual channel constraints [C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results. 2020: 25-28.

Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding [C]//Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019: 4171-4186.

Guan B, Wan Y, Bi Z, et al. Codeip: A grammar-guided multi-bit watermark for large language models of code [J]. arXiv preprint arXiv:2404.15639, 2024.

Kim J, Park S, Han Y S. Marking code without breaking it: Code watermarking for detecting llm-generated code [J]. arXiv preprint arXiv:2502.18851, 2025.

Kirchenbauer J, Geiping J, Wen Y, et al. A watermark for large language models [C]//International Conference on Machine Learning. PMLR, 2023: 17061-17084.

Koschke R. Survey of research on software clones [J]. 2007.

Lee T, Hong S, Ahn J, et al. Who wrote this code? watermarking for code generation [J]. arXiv preprint arXiv:2305.15060, 2023.

Li B, Zhang M, Zhang P, et al. Acw: Enhancing traceability of ai-generated codes based on watermarking [J]. arXiv preprint arXiv:2402.07518, 2024.

Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach [J]. arXiv preprint arXiv:1907.11692, 2019.

Mirsky Y, Demontis A, Kotak J, et al. The threat of offensive ai to organizations [J]. Computers & Security, 2023, 124: 103006.

Roy C K, Cordy J R. A survey on software clone detection research [J]. Queen' s School of computing TR, 2007, 541(115): 64-68.

Topkara M, Topkara U, Atallah M J. Words are not enough: sentence level natural language watermarking [C]//Proceedings of the 4th ACM international workshop on Contents protection and security. 2006a: 37-46.

Topkara U, Topkara M, Atallah M J. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions [C]//Proceedings of the 8th workshop on Multimedia and security. 2006b: 164-174.

Xiao Y, Chen Y, Ma S, et al. Decoma: Detecting and purifying code dataset watermarks through dual channel code abstraction [J]. arXiv preprint arXiv:2504.07002, 2025.

Yang B, Li W, Xiang L, et al. Towards code watermarking with dual-channel transformations [J]. arXiv preprint arXiv:2309.00860, 2023.