IS_FISAC_GROUP_3

by Varikuti MADHURIMA

V Semester B.Tech. (CCE)

ICT 3172: INFORMATION SECURITY

IMPLEMENTATION REPORT

KeySplitWatermark: Zero Watermarking Algorithm for Software Protection Against Cyber-Attacks

submitted by

VARIKUTI MADHURIMA	210953314
NALLAMILLI NAGA VENKATA REDDY	210953334
SANAKA JATIN KRISHNA	210953198
SAHITH MAGULURI	210953154

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Introduction

Cyber-assaults are evolving at a stressful rate. Incidents together with statistical breaches, ransomware assaults, malware infections, and phishing schemes have emerged as extraordinary. This evolution has delivered to the issues of software vendors and users wherein they need to save you a huge kind of attacks. Existing watermark detection solutions have a low detection rate inside the software. To address this problem, this paper introduces a modern-day blind Zero code-based completely Watermark detection approach referred to as Key Split Watermark.

The authors of this paper apprehend the restrictions of current watermarking techniques and advise a Zero-watermarking technique as a functionality answer. This method targets to enhance software program software protection against cyber-attacks and cope with the shortcomings of conventional detection strategies. The algorithm logically affords watermarks into the code making use of the inherent houses of code and offers a strong answer. The embedding set of regulations uses keywords to make segments of the code to supply a key-relying at the watermark. The extraction algorithms use this key to dispose of watermarks and locate tampering. Key Split Watermark is evaluated on tampering assaults on three unique samples with notable watermarks. The results show that the proposed technique evaluations promise effects closer to cyber-attacks which may be powerful and possible.

This research is mostly stimulated by way of the vital want to defend software programs from unauthorized access and manipulation, all whilst addressing the vulnerabilities that conventional watermarking methods might also pose. The paper significantly explores the fundamental concepts of Zero watermarking and the profound effect it can have on bolstering software program safety. To gain this overarching goal, the paper introduces a numerous array of algorithms, encompassing re-ordering algorithms, sign up allocation algorithms, unfold-spectrum algorithms, and different modern strategies.

Furthermore, the authors adopt a complete comparative analysis to evaluate the sensible effectiveness of the KeySplitWatermark approach within actual-world scenarios. This analysis encompasses numerous cyber-assaults and their outcomes while carried out to the covered software. By addressing capacity vulnerabilities and improving software program protection, the paper pursuits to contribute appreciably to the sector of cyber-security.

Methodology

Zero Watermarking:

It is a method that complements software safety through embedding watermarks within code. It utilizes inherent code residences to create key-based segments for watermark insertion.

Conventional Watermarking vs. Zero Blind Watermarking:

Conventional watermarking adds visible or hidden markings to data, aiding in ownership identification, it alters data, which can affect its integrity. In contrast, Zero Blind Watermarking, a type of zero watermarking, embeds imperceptible watermarks into data without altering its visual or structural properties, enhancing data security, and allowing for tamper detection and stronger security while remaining visually unchanged.

Algorithm Selection and Design:

The technique of zero blind watermarking is an advanced technique used to protect software programs without changing their functionality. It involves embedding and extracting watermarks with the use of unique algorithms to make certain safety. Here is an overview of the techniques and algorithms employed in this technique:

Embedding Algorithms:

These contribute to software program safety by means of securely putting watermarks or security capabilities into the code without compromising its functionality. They beef up the software's resistance to tampering and unauthorized get entry to.

Embedding Watermarks:

1.Re-Ordering Algorithms: These techniques rearrange code systems, making reverse engineering hard.

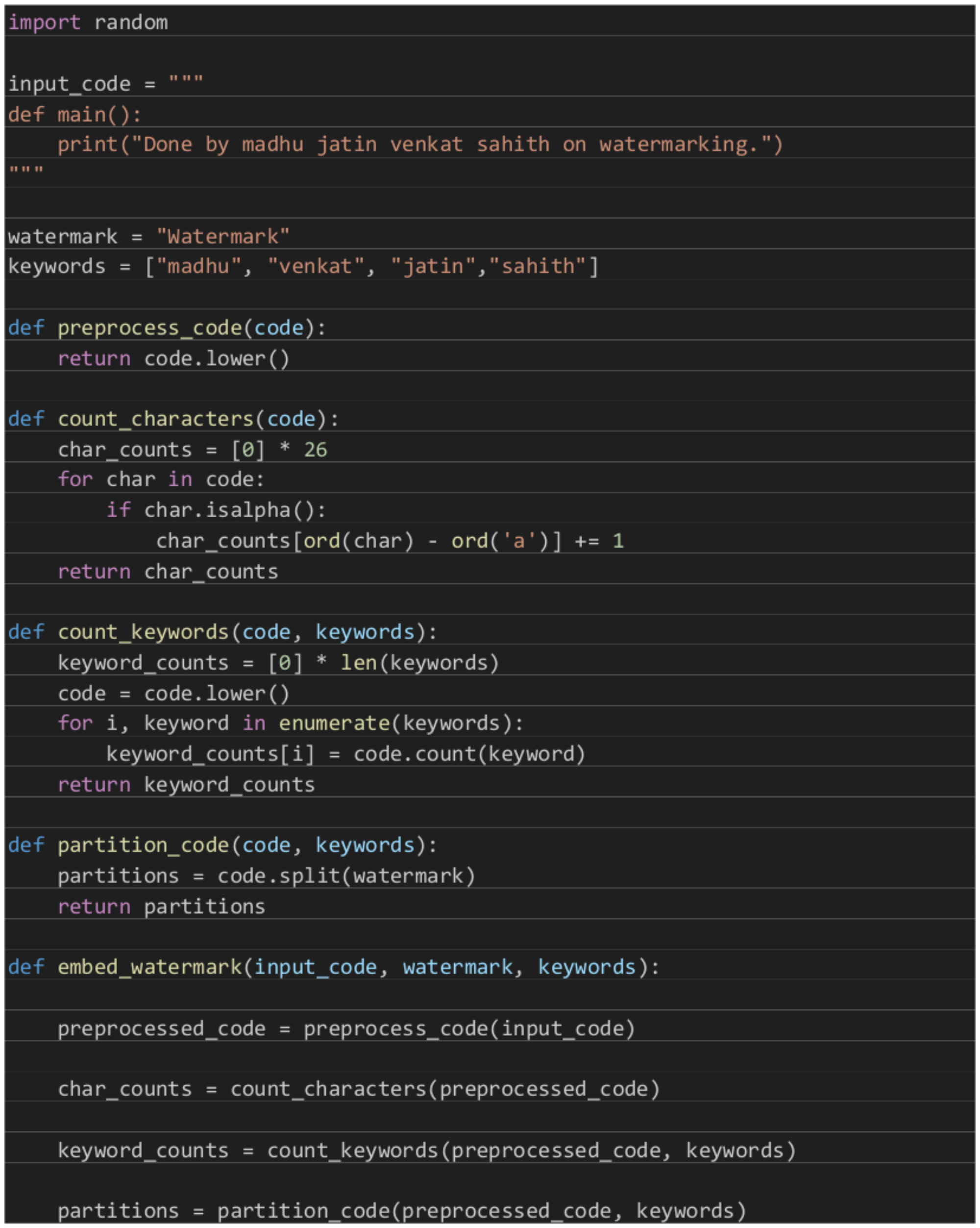
2. Register Allocation Algorithms: These optimize register usage for better performance and resilience against tampering.

3.Spread-Spectrum Algorithms: These disperse hidden watermarks throughout the code, making them difficult to get rid of.

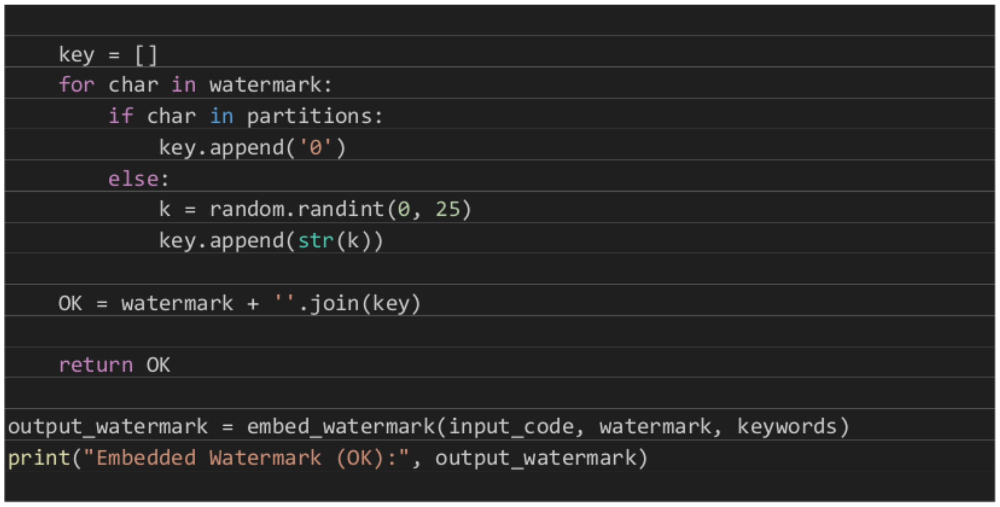
4. Opaque Predicate Algorithms: They introduce conditional statements with non-obvious consequences, obscuring code good judgment.

5. Code Replacement Algorithms: These alternative code segments avoid reverse engineering.

Here is the implementation and result of Embedding Algorithm (z26)

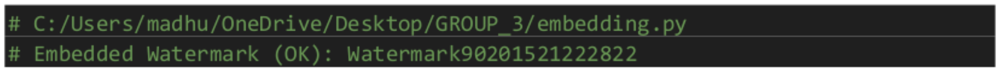


```python
import random

input_code = """
def main():
    print("Done by madhu jatin venkat sahith on watermarking.")
"""

watermark = "Watermark"
keywords = ["madhu", "venkat", "jatin","sahith"]

def preprocess_code(code):
    return code.lower()

def count_characters(code):
    char_counts = [0] * 26
    for char in code:
        if char.isalpha():
            char_counts[ord(char) - ord('a')] += 1
    return char_counts

def count_keywords(code, keywords):
    keyword_counts = [0] * len(keywords)
    code = code.lower()
    for i, keyword in enumerate(keywords):
        keyword_counts[i] = code.count(keyword)
    return keyword_counts

def partition_code(code, keywords):
    partitions = code.split(watermark)
    return partitions

def embed_watermark(input_code, watermark, keywords):

    preprocessed_code = preprocess_code(input_code)

    char_counts = count_characters(preprocessed_code)

    keyword_counts = count_keywords(preprocessed_code, keywords)

    partitions = partition_code(preprocessed_code, keywords)
```



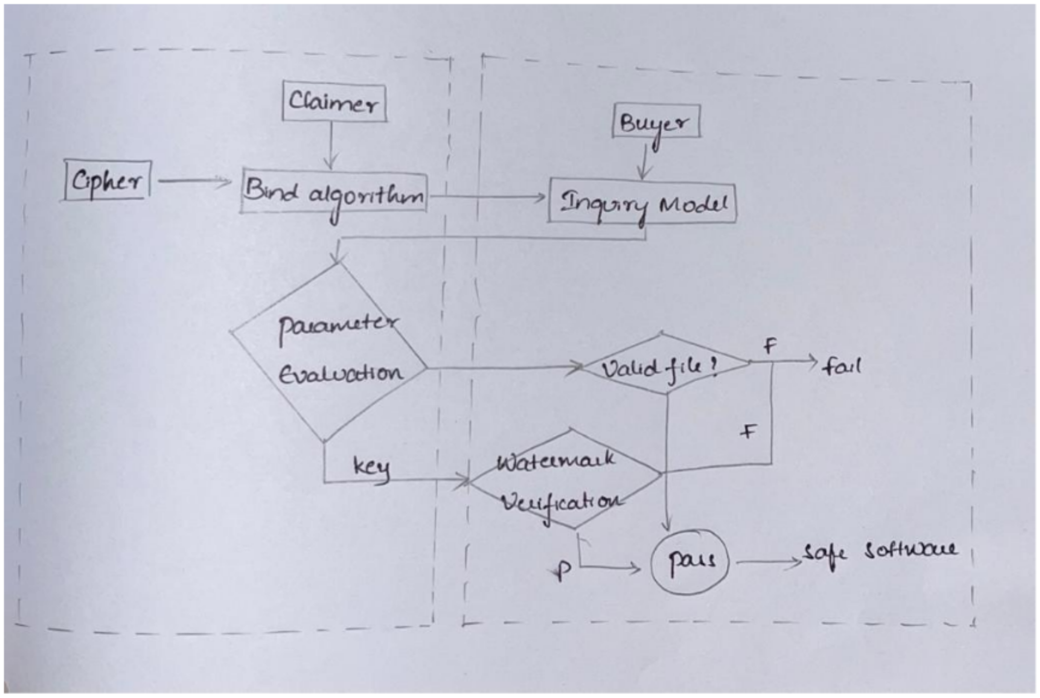
```python
    key = []
    for char in watermark:
        if char in partitions:
            key.append('0')
        else:
            k = random.randint(0, 25)
            key.append(str(k))

    OK = watermark + ''.join(key)

    return OK

output_watermark = embed_watermark(input_code, watermark, keywords)
print("Embedded Watermark (OK):", output_watermark)
```

RESULT:



```
# C:/Users/madhu/OneDrive/Desktop/GROUP_3/embedding.py
# Embedded Watermark (OK): Watermark90201521222822
```

Block Diagram of Proposed Approach:

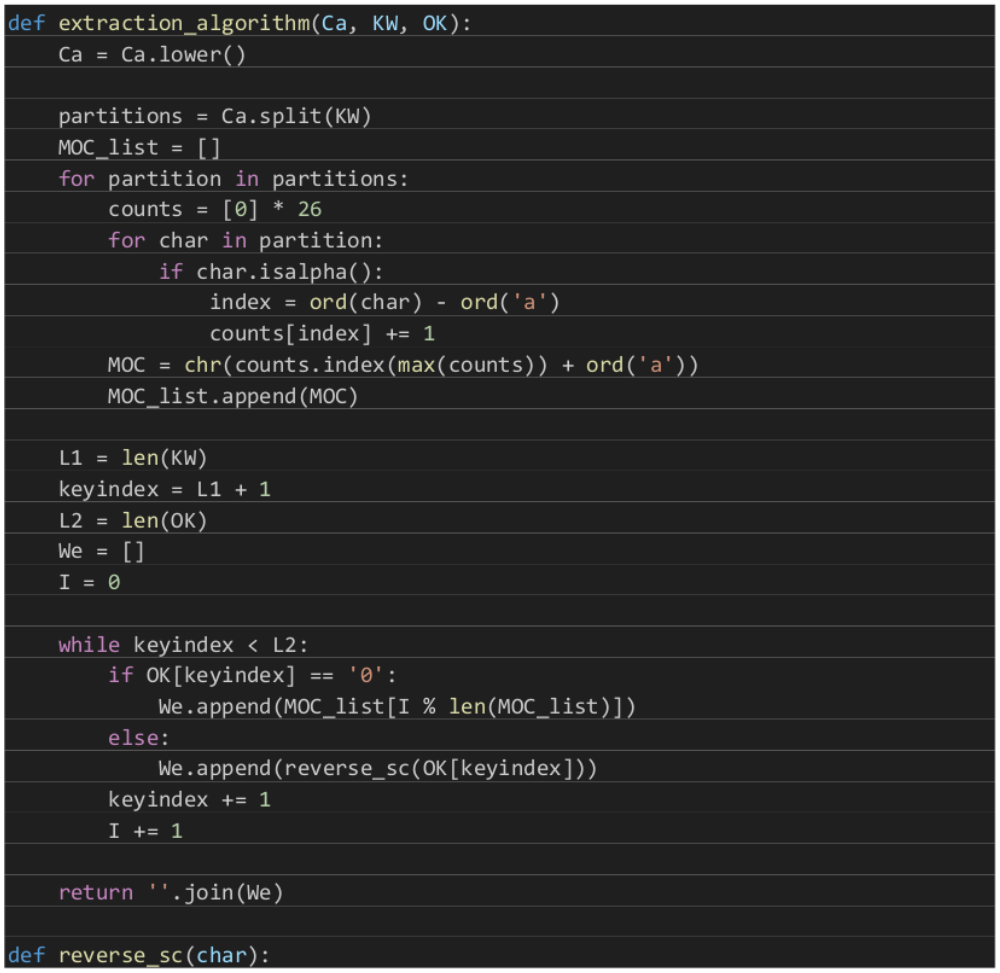


Extraction Algorithms: These supplement the safety method by ensuring the correct and secure removal of watermarks or security measures when important. They hit upon tampering and hold the software's integrity, improving overall security.

1. Dynamic Path and Graph-Based Algorithms: These screen code execution paths to come across tampering and unauthorized get right of entry too.

2. Abstract Interpretation Algorithm: This analyzes the code for vulnerabilities and strengthens the software's protection.

Here is the implementation and result of Extraction Algorithm



```
def extraction_algorithm(Ca, KW, OK):
    Ca = Ca.lower()

    partitions = Ca.split(KW)
    MOC_list = []
    for partition in partitions:
        counts = [0] * 26
        for char in partition:
            if char.isalpha():
                index = ord(char) - ord('a')
                counts[index] += 1
        MOC = chr(counts.index(max(counts)) + ord('a'))
        MOC_list.append(MOC)

    L1 = len(KW)
    keyindex = L1 + 1
    L2 = len(OK)
    We = []
    I = 0

    while keyindex < L2:
        if OK[keyindex] == '0':
            We.append(MOC_list[I % len(MOC_list)])
        else:
            We.append(reverse_sc(OK[keyindex]))
        keyindex += 1
        I += 1

    return ''.join(We)

def reverse_sc(char):
    k = 1
    reversed_char = chr((ord(char) - ord('a') - k) % 26 + ord('a'))
    return reversed_char

Ca = input("Enter input Ca: ")
KW = input("Enter the keyword KW (from OK): ")
OK = input("Enter the OK: ")

output_We = extraction_algorithm(Ca, KW, OK)
print("We (Extracted Text):", output_We)
```

RESULT:

```
C:/Users/madhu/Python/Python311/python.exe
c:/Users/madhu/OneDrive/Desktop/GROUP_3/EXTRACTION_ALGORITHM.py
# Enter input Ca: madhu jatin venkat sahith 1234
# Enter the keyword KW (from OK): keyword
# Enter the OK: keyword1357002468
# We (Extracted Text): fhjaaegik
```

The zero blind watermarking process includes embedding watermarks into the code without altering its appearance or functionality. Watermarks are invisible and imperceptible marks that are strategically located in the code. The key idea is to introduce those watermarks in a way that does not affect the software program's operation but affords a manner to discover any tampering or unauthorized get entry to.

Conclusion:

The KeySplitWatermark technique affords a groundbreaking approach to software program safety against cyber-assaults, addressing the evolving risk panorama and the limitations of traditional watermarking techniques. It offers a novel technique known as zero watermarking and contains a range of algorithms to decorate software program safety. Zero watermarking technique and set of rules selection lessen vulnerabilities. It strikes a balance between security and capability, making sure that the software stays completely operational.

Existing Software Watermarking Tools:

Many present software programs watermarking gear and techniques have been evolved and researched inside the field of cybersecurity. This equipment has provided treasured insights into unique methods for embedding and extracting watermarks in software programs.

Researchers have leveraged the understanding and techniques advanced through this equipment to create a much better and powerful software program safety mechanism in KeySplitWatermark.

Comparative Analysis with Existing Methods:

Prior research often includes comparative analysis of different software protection methods, including traditional watermarking techniques. By drawing from the strengths and addressing the weaknesses of prior tactics, KeySplitWatermark aspires to establish a much better defense in opposition to cyber-assaults.

Security Against Cyber-Attacks:

KeySplitWatermark complements software protection in opposition to cyber-attacks with the aid of introducing a Zero Watermarking technique. By integrating standards from security gear, it ensures the software's resilience in opposition to a large sort of cyber threats.

Software Security Tools:

By integrating elements from these software program security tools, KeySplitWatermark offers a higher protection against tampering and assaults. It makes it appreciably extra difficult for attackers to manipulate the software program at the same time as preserving the integrity of the software program's center capabilities. This technique strengthens software program protection in an evolving panorama of cyber threats.

Contributors:

VARIKUTI MADHURIMA

NALLAMILLI NAGA VENKATA REDDY

SANAKA JATIN KRISHNA

SAHITH MAGULURI

V.Madhurima

N.Naga Venkata Reddy

S.Jatin

[illegible]

Contribution Statement:

SANAKA JATIN KRISHNA:

Provided an introductory analysis of the KeySplitWatermark Technique, emphasizing its significance in the context of Cyber Attacks and Software Protection. And highlighted the advantages of zero watermarking for software security and pointed out the limitations of traditional watermarking techniques.

VARIKUTI MADHURIMA:

Described five different watermark embedding techniques, offering insights into fundamental embedding algorithms within the framework of KeySplitWatermark implemented the embedding algorithm (z26), and created the block diagram of the proposed method.

NALLAMILLI NAGA VENKATA REDDY:

Focused on enhancing the KeySplitWatermark approach by concentrating on the extraction algorithm. And introduced two specific types of extraction algorithms and implemented the extraction algorithm in Python.

SAHITH MAGULURI:

Conducted a comparative analysis of KeySplitWatermark with existing software security methodologies, discussed various software security tools, and concluded the analysis, elucidating the implications of the approach concerning software security and cybersecurity.

IS_FISAC_GROUP_3

ORIGINALITY REPORT

6%	6%	6%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	6%

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		