FORWARD CHAINING AND SELF-EMBEDDING WATERMARKING FOR TAMPER DETECTION IN A CONTINUOUS STREAM OF DATA

Sandip Hodkhasa and Huiping Guo

Department of Computer Science,
California State University, Los Angeles, California, USA

ABSTRACT

Watermarking is extensively used in various media for data transfer, content authentication and integrity. The continuous flow of data is always vulnerable to tamper. This research proposes a new watermarking scheme that detects tampering in a stream of data. The stream of data is dynamically divided into different sized groups using synchronization points. A computed watermark is embedded in each group by hashing the concatenating the current group and the next group. A secondary watermark is generated based on the current group that prevents tampering from any attacks in the current group. Watermark verification table is used to determine all possible scenarios for false results. Experiments are performed to show its efficiency. False results decrease as the group size becomes larger. Random burst attacked requires larger group size. The scheme also shows with the increase in grouping parameter 'm' which defines the synchronization point, the false positive rate decreases.

KEYWORDS

Cryptography, Digital Watermarking, Hashing, Information Hiding, Tamper Detection.

1. INTRODUCTION

Various applications that require continuous flow or streaming of data. This large flow of continuous data has applications in Internet of Things (IOT), Sensor networks and other IOT related fields [1]. Therefore, there is a constant need for data preservation, authentication, integrity and security. Research and development for security and privacy on such continuous streaming of data is exponentially growing. With new developments in various technologies, the exploitations such as vulnerabilities, complications, and loopholes in a flow of data also increases. Some of the issues include copyright infringement, data authentication, data integrity, illegal distribution, and others. [2] At the same time, data streaming over unreliable networks are subjected to data tampering and manipulation which is a concern as well [3].

Cryptography is one of the oldest forms of technology that is extremely used in data protection and security [2]. However, cryptographic algorithms can be computationally expensive because of their modular exponential multiplications and power; hence it is not widely applicable or used in Wireless Sensor Networks (WSN) and IOT [4]. Digital watermarking is a new form of technique that "complements cryptography and steganography" [5] and aids in data integrity, authentication and protects against illegal copying and tampering of data. In digital watermarking, the data is embedded into the host media (such as video, audio, images, and text) just as in steganography, however, in steganography the host is embedded with a secret message whereas in watermarking the host contains several types of meta data such as its ownership, origin etc. Watermarking is computationally a lightweight solution as opposed to cryptographic algorithms that require multiple iterations of modular calculations. Hence making watermarking a great candidate for WSN, IOT and similar type of applications.

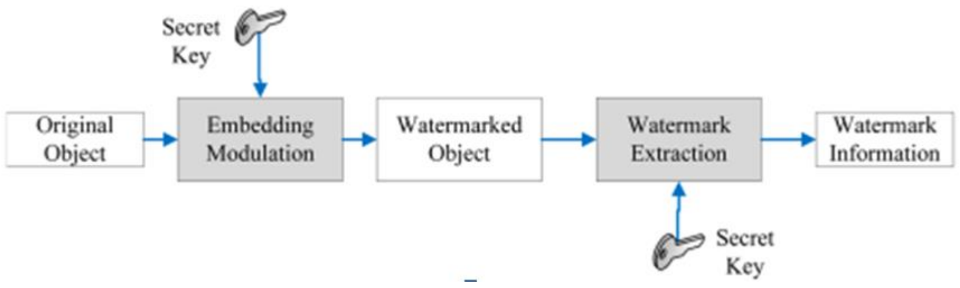

Figure 1. A watermarking technique [6]

Digital watermarking has the following major requirements [7]:

1. Transparency: A Secret data embedded into the host media that is invisible to plain sight.
2. Robustness: Watermarking should handle various attacks and cannot be easily destroyed.
3. Security: Embedded watermarking cannot be removed from the host. Removal of the watermark can lead to destroying or demolishing the host data as well.
4. Payload/Capacity: Overhead capacity needed to embed into a watermark. Embedding watermarks requires some memory.

The usage of digital watermarking provides the following [8] advantages:

1. The computation required to generate a watermark and embed into the host data is typically very lightweight. Hence it requires low consumption of energy compared to its peer technologies.
2. Since the host can embed the watermark into itself, there is low overhead for communication. This is very advantageous in WSN and IOT communications.
3. Lack of encryption and decryption algorithms, and single computation for most watermarking generation reduces end to end delay in network communication.

Digital watermarking can be categorized into the following branches [9] [10] [11]:

1. Fragile: Watermarking that is extremely sensitive to modifications and can sense slightest change in the watermarked data. It is useful for author authentication of digital content.
2. Robust: Such watermarking can endure various voluntary or involuntary attacks that can cause data manipulation. Attacks on host media carrying robust watermarking typically degrades the host media. Therefore, robust watermarking is extremely useful in illicit copying of media.
3. Semi-Fragile: These types of markings make use of both the above-mentioned schemes. Semi-Fragile watermarking can locate the location of tampering as well as the algorithm can restore the watermark if tampered.

This current research makes use for fragile watermarking to detect tamper in a continuous stream of incoming data. This source of incoming data can come by myriad applications from online websites to sensor network such as stock market to various types of sensors installed in smart cities and forest for temperature detection. In this research we assume there is continuous flow of data. This technique is applicable to other types of data, however, this research uses only integers for simplification.

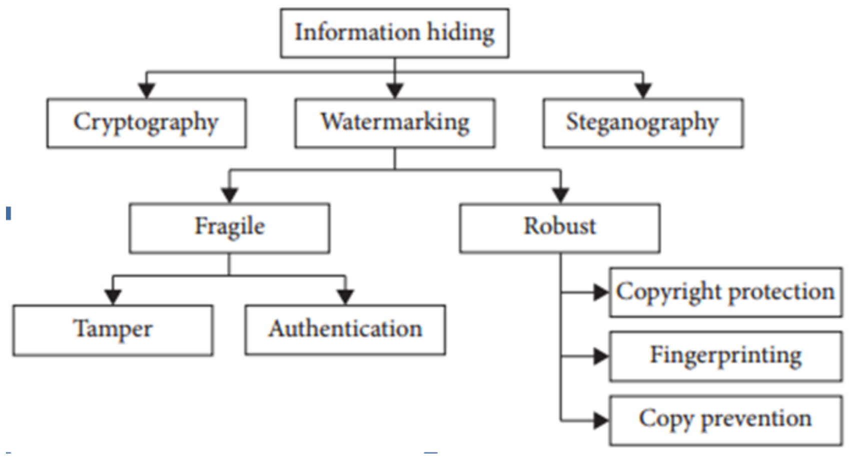

Figure 2. Common information hiding techniques [5].

In chapter 2, previous related works are reviewed and their flaws, whereas in chapter 3, the proposed algorithm is discussed. Primary and secondary watermark is introduced and the process of embedding, and extraction of the watermarks is also discussed. Table 1 shows the water detection table and Table 2 summarizes the list of all symbols used. Chapter 4 talks about the generation and the use of synthetic data. It also mentioned about the false positive and false negatives values that can get generated using this approach. Watermarking detection rationale is also introduced. The watermark rationale table discusses how false positive and negative values are calculated. Also, pros and cons of using the two watermarking scheme is considered. Different types of attacks are conducted, and the results of such attacks are examined and reviewed to deduce conclusions. The conclusion is discussed in chapter 5.

2. Related Work

The communication using text and numbers requires least amount of bandwidth among other modes of communications and yet can be one of the most complicated media to be watermark. [12] There are various techniques to watermark text and integers. This current research work is a very special type of text communication: a continuous flow of streaming data that is comprised of integers. Nonetheless, the algorithm mentioned in this research can be effortlessly revised to watermark text and other media. There are few methods that can be applied in watermarking such as [11] spatial domain techniques, and frequency domain technique. Spatial domains consist of least significant bit (LSB) coding which is the technique used in this research.

It is perceived that that Guo et al. [13] were the first to use fragile watermarking scheme to embed watermarking in a continuous flow of data. Their scheme chains group of inflowing data and embed watermark into it. In their scheme they propose a technique of dynamically grouping data which ends with a synchronization point which is a specific data element. This data element is defined by a hash computation of the data element whose value mod ‘m’ is zero, where ‘m’ is a number kept secret and only known to source node and sink node or end clients. Hash values are calculated using the secure hash function of each element in the group and then of the groups. Then a hash function is again performed of two consecutive groups. The hash values are then used to embed into the data.

A lightweight chained watermarking (LWC) scheme was introduced [14], where they improved upon [13] by performed less regressive of hash computation and tried address few issues by improving on computational overhead. In [4] FWC-D scheme was proposed. The authors added a group delimiter instead of grouping dynamically based on synchronization point.

[15] used a dual-marking fragile watermarking system based on the character. They also dynamically grouped data and embedded chained watermark. In their algorithm they used blank spaces as the watermark which generated from the characters that were converted from numeric data type.

[14] does not account for any false positive or false negative values. In [15] false positive rate is still too high and there seems to be no evaluation for the false positive rate with group size or for false negative rate. In [13] there is a trade-off between the security and precision. Most of the above-mentioned work does not account for small sized group and have high embedding and extraction time. It seems imminent that there is a requirement for a new algorithm that can provided integrity and remain tamper proof for all data elements within a group.

3. Proposed Algorithm

This proposed algorithm uses fragile watermarking system where watermark is embedded into the dataset, the dataset will experience some sort of minor distortion. This work also employs watermarking in the least significant bit scheme. The embedded algorithm is simple to implement and able to handle large sets of data in relatively short time.

3.1. Grouping and Synchronization Point

Assume that there is an endless and continuous flow of data. After the data is sensed by a sensor, this algorithm should be applied and then send to the receiver. The sender will have to form two groups and store data into the two buffers. Each incoming data element is filled into a group, the size of which is determined by synchronization point using following equation:

$$H i \bmod m == 0 \qquad \text{Equation 1}$$

Where H is a cryptographic hash function such as MD5 and SHA for each element s_i. Such hash function takes a variable length string as an input with a key and returns a fixed length hash value of the string. As each incoming data element goes through a hash function, equation 1 is performed to check if a data element is a synchronization point. Once the synchronization point has been encountered, that data element marks the end of that current group, and the buffer ends there forming one group. Similarly, second group is formed using the hash function as mentioned above and the list of data elements are saved into a second buffer. A set S is defined by a list of elements s_i, such as $S = \{s1, s2, s3, s4...\}$. Correspondingly, H_i denoted the hash value of each element s_i. It is to be noted that a synchronization point is dependent on the value of '*m*'.

Another factor that drives the size of the group is '*L*' defined as lower bound of a group size. Lower bound is the minimum size of the group that is required to form a group. Even though a synchronization point is encountered, if the size of the group is lower than '*L*' then the algorithm continues to find another synchronization point. As the two groups are formed. It is later analyzed how the value of '*L*' is critical to the authentication of the data at the receiver's end. However, as the range of the group grows up to 'U' - upper bound, (or the maximum value the group size can be) then the group ends.

Algorithm 1: Creating List of Data

```
1.  Create List1, List2
2.  List1 = dataGenerator (int $m$, $L$, $U$)
3.  List2 = dataGenerator (int $m$, $L$, $U$)
4.  Function: dataGenerator (int $m$, $L$, $U$) → List list
5.  while (True):
6.      number = generate a random number between a range
7.      subList.append(number)
8.      hashValue = $H$(number)
9.      return subList if(hashValue % $m$ == 0) and size(subList) > $L$
10.     return subList if(size(subList) == U)
```

3.2. Watermark Generation and Embedding

Once the two list is created, they are identified as two groups: current and next group, as *currentGroup* and *nextGroup*. In the *currentGroup* all the data elements of the *List1* are concatenated., whereas, in the *nextGroup* all the data elements of the *List2* are concatenated. The data elements in *List2* are the data elements that follows data elements in *List1*. Now two watermarks are generated involving the two groups as follows in Algorithm 2 and 3. As the watermark is embedded that data is then sent to receiver.

Algorithm 2: Create Group Hash

```
1.  Function: CreateGroupHash(List: list1, list2) → List: Hash1, Hash2
2.      currentGroup = {$s_1||s_2||s_3$…}
3.      nextGroup = {$s_n||s_{n+1}$ …}
4.      //When getting hash values the last two bit of each data element is ignored
5.      Hash1 = $H$(currentGroup||nextGroup||key1)
6.      Hash2 = $H$(currentGroup||key2)
7.  return Hash1, Hash2
```

Algorithm 3: Watermark Embed

```
1.  Function: WatermarkEmbed(Hash1, Hash2)
2.      for each data element 'i' in List1:
3.          replace last bit of 'i' by last bit of Hash1
4.          replace second last bit of 'i' by last bit of Hash2
5.          move right to the next element in List1
6.          move left to the next bit in Hash1 and Hash2
7.      end for
```

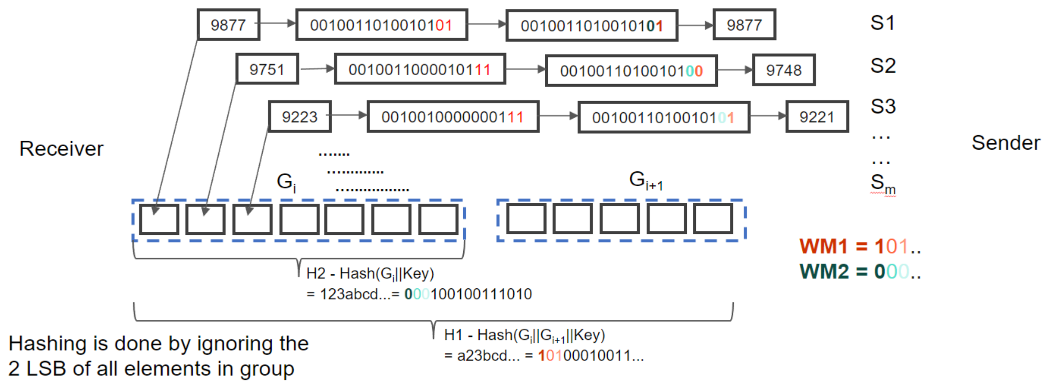

Figure 3. A graphical representation of the embedding process

3.3. Watermark Detection

As the data is received from the sender at the receiving end, the watermark extraction process is implemented. The extraction and the embedding processes are very similar. As the data is being obtained by the receiver, sink or the server, the incoming should be added to the buffer or list as it was done in *List1* and *List2* algorithm 1. Two sets of lists are created that terminates at the synchronization point using equation 1. Just to note that these data already contain the two watermarks. Once the lists are formed, algorithm 2 can be used again to generate the hash values of the two lists. Also point to be noted that when the hash values are generated, the algorithm ignores the last two bit of each data elements and then concatenates each data element to the group. Once the hash values are generated, two watermarks are created as *WMG1* and *WMG2*. *WMG1* which is the primary watermark is generated from *Hash1* by concatenating the extreme right 'k' bits ('k' being the size of the current most group or list of data received). Similarly, like *WMG1*, *WMG2* can be generated from *Hash2* which is referred to as secondary watermark.

Correspondingly, using the new lists that are created at the receiving end, *receiveList1* and *receiveList2*, the embedded watermark is extracted. This extracted watermark is then compared to watermark that is generated using the data received at the receiving end mentioned in the previous paragraph. The watermark that is extracted by extracting the last bit of every data element (which is also the watermark generated by getting the hash value of *H* (*currentGroup*||*nextGroup* ||*key1*)) is referred to as the primary watermark or *WME1*. The watermark that is extracted by extracting the second last bit of every data element (which is also the watermark generated by getting the hash value of *H* (*currentGroup*|||*key2*)) is referred to as the secondary watermark or *WME2*.

With the availability of two watermarks, the verification for integrity and modification makes it less complicated and easy. We use *WME1* to check the integrity of the data, it checks if a group of elements have been deleted, while *WME2* is used to check if there is a modification in a group. Two types of verification will be done using *WME1*: preliminary and final verification. Preliminary verification is done using the *WME1* for the current and next group, while final verification is based on the preliminary and final verification of previous group and current group using *WME1*. *WME2* will be used when the final verification of the current group is turned out to be false or the *WME1* and *WMG1* are not matched.

3.4. Watermark Verification

To verify the integrity of the group of incoming data, two buffers or lists of data are constructed as mentioned above. Since there are two groups of data, we have two types of verification: preliminary and final denoted as *pV* and *V* respectively, and two watermarks are present. The verification can be done using algorithm 4 as shown below. The preliminary and final verification of the previous group is assigned as *pV0* = False and *V0* = False in the beginning. After each iteration, the value of *pV1* and *V1* is assigned to *pV0* and *V0*, where *pV1* and *V1* are the preliminary and final verification of the current group. *V2* is the verification result of comparing the secondary watermark embedded and generated at the receiving end.

Algorithm 4: Watermark Verification

```
1. V2 = True if (WME2 == WMG2) else V2 = False
2. if (WME1 != WMG1):
3.     pV1 = False
4.     V1 = V0 & pV0
5. else: V1 = pV1 = True
```

This current work makes some assumptions and takes leniency in the data. When the watermark is being embedded and extracted, each data element of group set goes through a small distortion i.e. the last two bit of each data element is ignored while generating watermark, and the watermark once generated is embedded into the last two bit of each data element. Watermark one can also we called as the integrity watermark and watermark two is also called the anti-modification watermark. The watermark detection and verification are based on the following Table 1. For cases 1-4, where the watermark is matched it means that the group that is modified or if a group is missing, that has been successfully verified. Rest of the cases are discussed below in Table 1.

Table 1. Watermark verification table

Predicates	**Cases**	**Previous Group**		**Current Group**		**Current Group**
		Group 1 - G1		Group 2 - G2		
		PV0 (WM1)	V0 (WM1)	PV1 (WM1)	V1 (WM1)	V2 (WM2)
Watermark Match	1	TRUE	TRUE	TRUE	TRUE	
Yes	2	FALSE	FALSE	TRUE	TRUE	
		Entire group between previous and the current group may be absent				
	3	FALSE	TRUE	TRUE	TRUE	
		Entire group between previous and the current group are absent				
Watermark Match	4	TRUE	TRUE	FALSE	TRUE	
No	5a	FALSE	TRUE	FALSE	FALSE	FALSE
		G2 Modified				
	5b	FALSE	TRUE	FALSE	FALSE	TRUE
		Initial False Positive, but WM2 confirms group missing between G1 and G2				
	6a	FALSE	FALSE	FALSE	FALSE	FALSE
		G2 Modified				
	6b	FALSE	FALSE	FALSE	FALSE	TRUE
		G1 and G3 modified, Groups missing				

Case 5a: In case 5a, if the preliminary verification and the final verification of the current group is false, it means that either the current group or the next groups is modified, i.e., WM1 did not match. Back checking with the previous group it turns out that the previous group is verified to be true. Therefore, at this point, WM2 is used to verify the authenticity of the current group. If the WM2 is false, that only means the current group is modified.

Case 5b: In case 5b, if the preliminary verification and the final verification of the current group is false, it means that either the current group or the next groups is modified, i.e. WM1 did not match. Back checking with the previous group it turns out that the previous group is verified to be true. Therefore, at this point, WM2 is used to verify the authenticity of the current group. If the WM2 is true, that only means the current group was not modified. This results in conclusion that there must be a group missing between G1 and G2

Case 6a: In case 6a, if the preliminary verification and the final verification of the current group is false, it means that either the current group or the next groups is modified, i.e., WM1 did not match. Back checking with the previous group, if it turns out that the previous group is verified to be false, then it means that at this point, either there is a group missing between the previous and the current group or there is a group missing between the current group and next group. WM2 is used to verify the authenticity of the current group. Since the WM2 is false that only means the current group is modified.

Table 2. List of Symbols and its meaning

Symbol	**Meaning**
H	Cryptographic Hash Function
s_i	Data Element
S	Data Set Element
m	Synchronization Point
L	Lower bound of group size
U	Upper bound of group size
List1	all the data elements of *currentGroup*
List2	all the data elements of *nextGroup*
WMG1	the primary watermark is generated from Hash1
WMG2	the secondary watermark is generated from Hash2
WME1	primary watermark extracted from the received list
WME2	secondary watermark extracted from the received list
PV0	Preliminary Verification for previous group using WM1
V0	Final Verification for previous group using WM1
PV1	Preliminary Verification for current group using WM1
V1	Final Verification for current group using WM1
V2	Verification for Watermark 2
WM1	Watermark 1, based on the grouping of current and next group
WM2	Watermark 2, based on the current group only

Case 6b: In case 6b, if the preliminary verification and the final verification of the current group is false, it means that either the current group or the next groups is modified, i.e., WM1 did not match. Back checking with the previous group, if it turns out that the previous group is verified to be false, then it means that at this point, either there is a group missing between the previous and the current group or there is a group missing between the current group and next group. WM2 is used to verify the authenticity of the current group. Since the WM2 is true that means that current group is not modified, which means that either the previous and next group is modified or there are groups missing.

4. Experiments

Synthetic data is used for a controlled experiment, an infinite flow of data, *S* is generated using a standard random function generator. Data ranging from 9000 to 9999 is generated as integers. Data is uniformly distributed between the range. The key values such as group separator - *m*, the two key *k1*, and *k2*, lower bound of the group *L*, are decided and assumed to be kept secret. The algorithm is tested with a minimum of 10000 data elements. The system specifications for the simulation are Windows 10 Pro, Intel(R) Xeon(R) CPU E5-1620 0 @ 3.60GHz.

4.1. Attack

The watermarking model was tested by various types of attack. The attacks were also randomly generated based on the Poisson's distribution. The random Poisson's distribution function was implemented, and it generates a random number. Based on the number generated, the following attacks were performed using the modulus of the iterations:

```
If (iteration % x == 0):
        "Modifying by appending"
Else if (iteration % x == 1):
        "Modifying by Deleting"
Else if (iteration % x == 2):
        "Modifying by Modification"
Else if (iteration % x == 3):
        "Modifying by elimination a group"
```

Where ‘*x’* is the random value that is generated using the Poisson’s distribution. This random value ‘*x*’ is an average of a list of ten random numbers and then used in the experiment. The *iteration* is a counter that runs and counts the random data element that is being generated. The attacks that are performed are based on the *iteration* and *x* as following:

Modifying by Appending: As the groups are formed, the groups are modified by adding data elements into the group. Random data elements were added into the current group post watermarking is embedded. After modification, the group is then sent to the receiver.

Modifying by Deleting: Once the group is formed and watermarking is performed into the data set, a data element is randomly picked from the current group and deleted from the group reducing the size of the group. The modified group is then sent to the receiver’s end for verification.

Modifying by Modification: In this case, modification is done by taking the data set and picking a random element within the current watermarked data set and modifying that value of the data element by some type of operations. In this case we modified a data element by adding four to the integer. Issues with modifying the data element by a value less than four is discussed later.

Modifying by Elimination in a group: In this situation of modification, the modification is done to the entire group. As the data set or group is watermarked, after being watermarked, the entire watermarked group is substituted but another set of random data elements. This new set of random data elements may or may not be of the same size.

4.2. False Values

4.2.1. False Negative

False negative is when the data is modified after watermarking however, the algorithm for watermark detection doesn’t recognize the data tampering or data modification. This experiment uses different values of lower bound groups and later results are discussed based on different sizes of the groups.

4.2.2. False Positive

False positive is when the data is not modified after watermarking, however, the algorithm for watermarking detection recognizes the data set as being tampered or modified. This experiment also accounts for data showing false positive for different sizes of lower bound groups. The following sections discusses different scenarios and reasons where false flags are raised.

4.3. Least Two-Bit Modification

Least two-bit modification of the data is an important modification which dictates the verification of the watermark that was embedded. When making a modification in the last two bit of the data, the watermarking is performed by ignoring the last two bit of the group data, when the data is watermarked, and if there is an attack where the attacker adds a value from one to three, such a modification remains un-noticed. The following example gives a better idea.

Hash is calculated ignoring the last two bit.
Ex: 9726 becomes 9724, when ignoring the last two bit.

$$\text{Therefore, } H(9724) = H(9725) = H(9726) = H(9727) \qquad \text{Equation 2}$$

If after watermarking, there is a modification made by the attacker which only changes the least two bits, then watermarking verification will remain undetected. Hence, this algorithm can be used only where slight modification is acceptable.

4.4. Matching Extracted Watermark

There are instances where the post modification in a group of data set, or modification in the entire group of data still results in same watermark. Since the watermark is same, this results in false negative, where the data is modified but the algorithm is not detected. This instance typically occurs for low-sized groups. The subsequent is an example of such a case, the size of the group here is 5:

DATA FROM SENDER

Original Data:	[9759, 9785, 9738, 9040, 9776]	
Group + Key:	97569784973690409776 || 11	(Ignoring the last 2 bit)
Hash Value:	7468bd11bcf7572a0066ec78efc139ca	
Embedded Data List:	[9757, 9786, 9738, 9042, 9777]	
Embedded Data List:	[9761, 9786, 9738, 9042, 9777]	(After Attack)
Hash Value in bits:	74…. = 0111 0100 …	
Watermark to Embed:	011101	

DATA TO RECEIVER

Received Data:	[9761, 9786, 9738, 9042, 9777]	
Group + Key:	97609784973690409776 || 11	(Ignoring the last 2 bit)
Hash Value:	75e1e7e7789b10398d3a91152edc4876	
Embedded Data List:	[9757, 9786, 9738, 9042, 9777]	
Embedded Data List:	[9761, 9786, 9738, 9042, 9777]	(After Attack)
Hash Value in bits:	75…. = 0111 0101 …	
Watermark to extract:	011101	

Even though the data set has been modified, the watermark that is inserted and extracted are same. It can be argued that because the hash function used in this algorithm is MD5, and MD5 is not a great security hash function, there are hash valves which have similar values. However, the next section tackles other secure hash functions. It is important to note that, for small groups, there can be several false negatives.

4.5. Lower Bound Group Size

Since, in the previous section it was determined that the low-sized group plays a crucial role in determining the rate of false negative or positive values, the following graphs show the decreasing of the false values as the group size increases. It can be seen in the Figure 4, as the group size increases the false negative and positive values decreases. Two hash functions were used, MD5 and SHA256. The hash values obtained were then used as forward and backward embedding.

In forward embedding, the bits from the beginning of the hash values are used for watermarking, whereas in backward embedding, the trailing bits are used for watermarking the current group of data. Using both methods, with the increase in group size, the false results decreased. The false positive and false negatives both converge to zero at the same time as the lower limit of the group size becomes 10. In

Figure 5, there is a direct comparison of the two functions shown with forward and backward embedding. There is typically not a major difference. They both follow a similar path and pattern, with bottoming out at the lower bound of group size of 10.

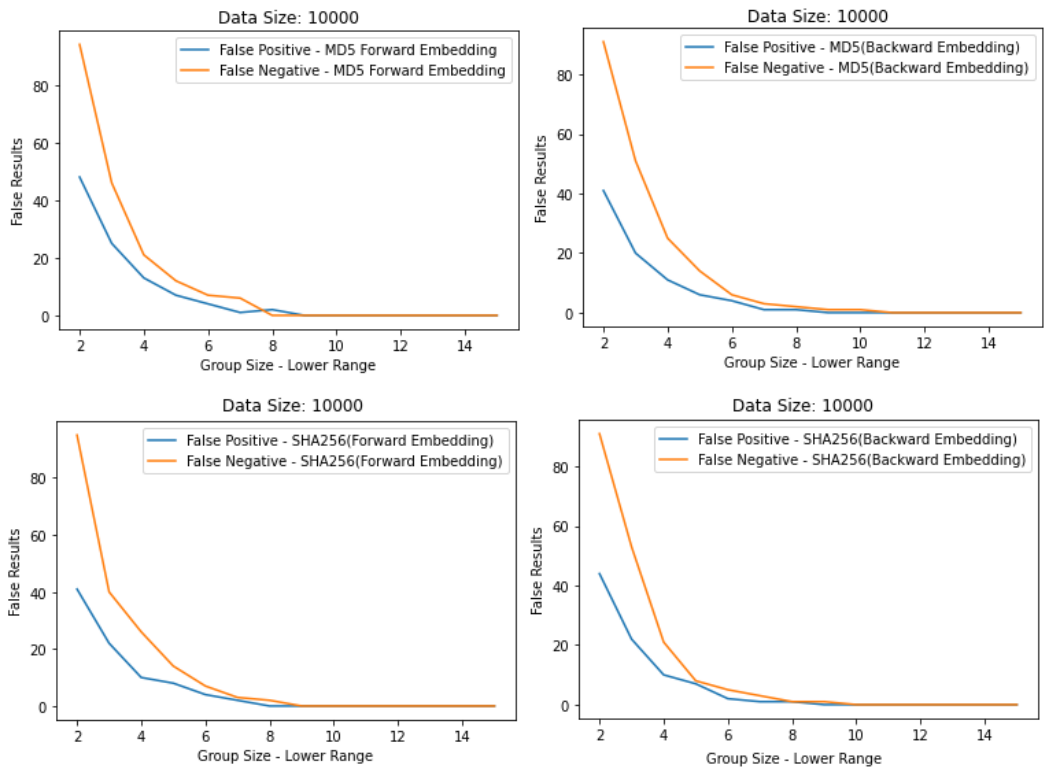

Figure 4. MD5 and SHA265 false positive and false negative values versus lower bound group size for a data size of 10000

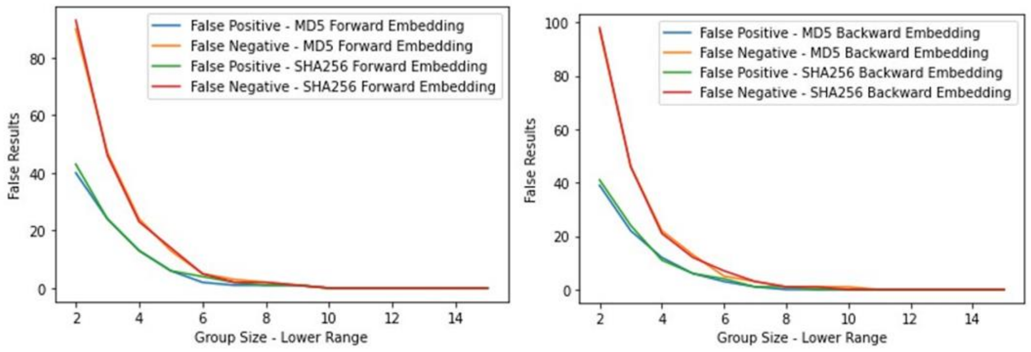

Figure 5. MD5 and SHA256, Forward Embedding and Backward Embedding comparison

4.6. Watermark Scalability

Figure 6 shows the watermark scalability.

Figure 6 shows as the data size increases, the time required for watermark embedding and extraction also increases. It can also be seen that the embedding and extraction time frames are different for different group sizes. While a group size of 10-20 takes the least amount of time, a group size of 50-60 takes the maximum amount of time. The time calculated in

Figure 6 is the time required to embed plus the time required to extract the watermark.
Figure 6 shows the scalability of the size of data with respect to the size of group and the false results. As it can be seen, with high data sizes the false result increases, however they decay as the lower bound of the group size increases. It can be clearly seen that regardless of the data size, all of plot lines converges to zero false results when the group size approaches to 10.

4.7. Burst Attack

Burst attack was performed in this model in which a modification is repeated 'μ'times also known as the burst attack length. In this attack, insertion, deletion was done at equal probability and applied randomly using Possion's distribution function. The attack is done multiple times within the same group and pattern has been observed. The attack was performed with:

1. Random Size 'μ' - a random number was generated and assigned to μ each time and the insertion or deletion was done μ times.
2. Fixed Size 'μ' – The insertion or deletion was done for four iterations, and in each iterations the attack was incremented by one.

As it can be seen in

Figure 7, the false positive rate (false positive over data unaffected) falls as the group size increases. Small groups tend to have high false positive rates as small groups have high false positive. The false positive rate decreases with increase in group size which also means the false positive rate decreases as well. It seems that when that attack is random, the algorithm requires a high lower bound compared to the previous attacks that was performed.

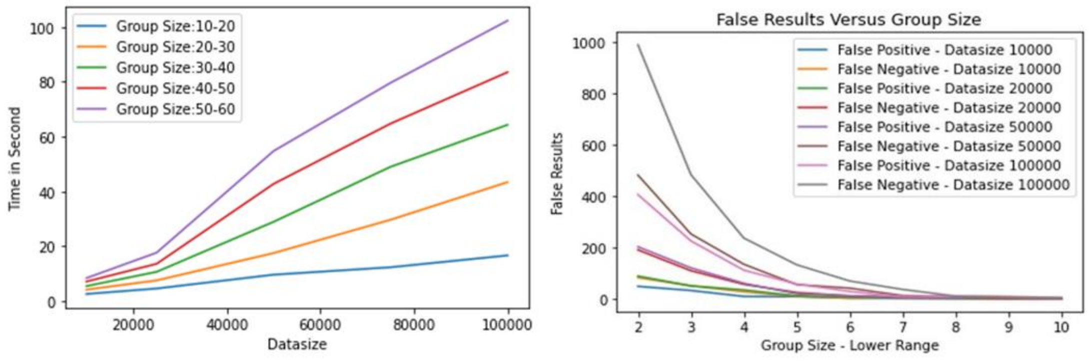

Figure 6. Scalability of watermarking shown with different group size and data size.

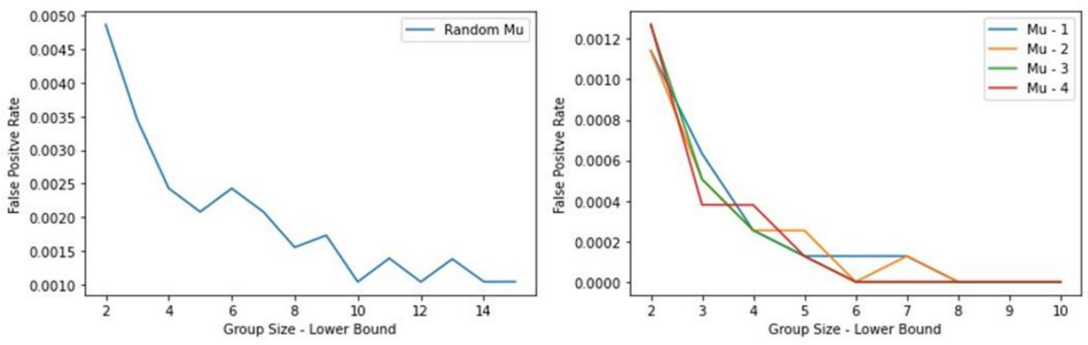

Figure 7. False positive rate versus lower bound of group size with μ varied randomly and with μ increasing from 1 to 4

4.8. Group Attack

The final attack was performed to measure the false negative rates when incoming group or groups were deleted. As the incoming groups are forming, the attack deletes the next group(s) that are being formed. The number of groups to be deleted depends on random number based on the Poisson's distribution. It was observed that the false negative values were zero and the algorithm detected all the modifications that were being made during the inflow of the data.

4.9. Grouping Parameter – 'm'

In this watermarking scheme, 'm' defines the synchronization point. which is used as a parameter to form groups. The size of the group partially depends on 'm'. The algorithm ensures that the size of the group is maintained within the range of the group that is bounded by lower and upper bound.

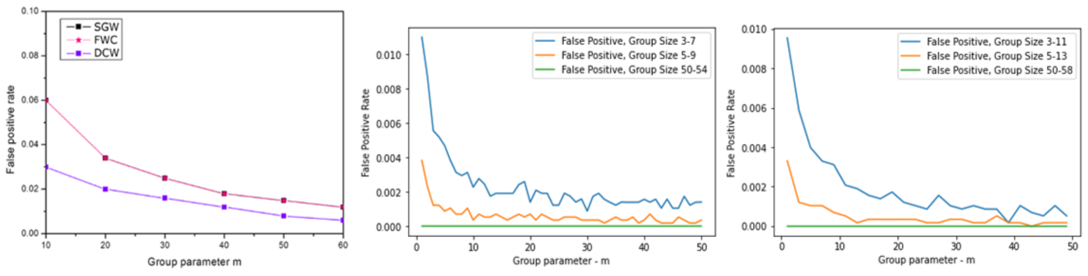

Figure 8 (Left image) is taken from [15] where they compare the group parameter and the false rate. Whereas, in

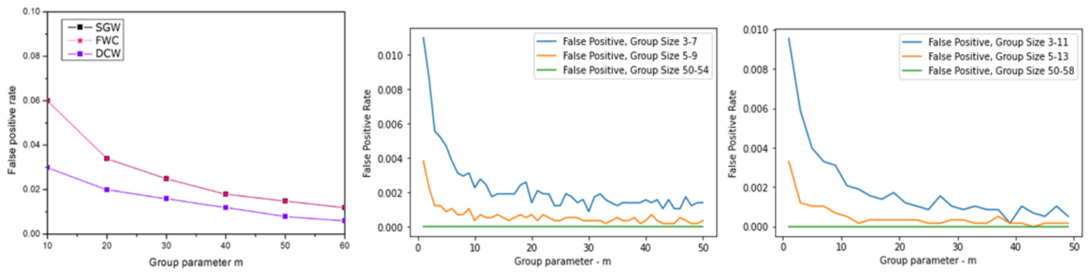

Figure 8 (Right and Middle image) it can also be seen that the false rate decreases as the group parameter increases. In [15], the lower bound 'L' is 50, whereas in this experiment the value for 'L' is very low for which the false rate tends to zero.

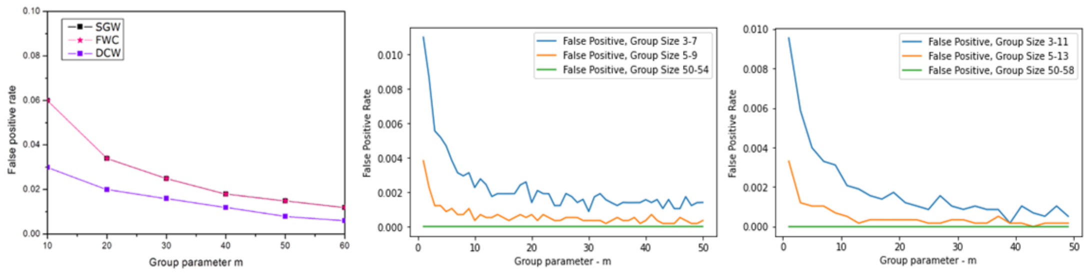

Figure 8 (middle and right image)**Error! Reference source not found.** shows two images for the same parameters – group parameter *m* and false positive rates, however the group range is different. For the image in middle the group range is 4 and the right plot the range is 8. Also, it must be noted that for small group sizes the false positive rate tends to zero as the group parameter 'm' increases.

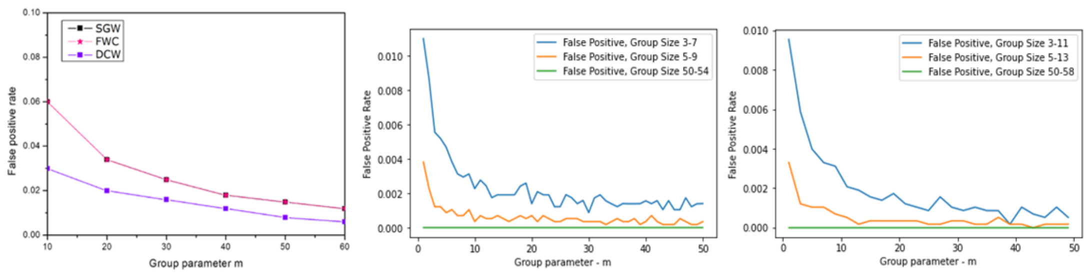

Figure 8. (Left) Results taken from [15] which compares false positive rate with group parameter, (Middle and Right) false positive versus group parameter for different range of lower and upper bound

5. CONCLUSION

As it can be seen, this algorithm has established that with a minimum of a group size of 10, data elements have successfully shown zero false results. When groups are sized under ten, false positive and false negative results are generated. Therefore, an application will have to alter data the group size to this threshold size in order to maintain the data integrity Another consideration to be noted that the application also has be adjusted so as the application can tolerate small distortion of data. Because this algorithm embeds the watermark by modifying that last two bit of data, the data is slightly modified. Also, it needs to be noted that when modifying the data i.e., tampering a data element, altering the value by less than three will make the data undetected. In other words, the alteration of the data elements needs to be more than three for the algorithm to detect any tamper. However, if the sensor can tweak the data in such a way that the least two significant bits does not impact its use in that application, then the algorithm can successfully maintain its integrity as well as prevent various types of attack. The current model used two different types of hash functions, and when the watermark was forward embedded and backward embedded, it produced similar results. They emit same false positive and false negative and converge to zero false results around the same lower bound of group size. In this research, the algorithm successfully mitigates false positives and negatives while maintaining the integrity of the data. The experiments successfully link the data that are formed into groups and watermarks can be embedded into the groups with small distortion. With slight modifications, the algorithm can be used successfully in various applications.

REFERENCES

[1] X. Sun, J. Su, B. Wang and Q. Liu, “Digital Watermarking Method for Data Integrity Protection in Wireless Sensor Networks,” International Journal of Security and Its Application, vol. 7, no. 4, pp. 407-16, 2013.

[2] I. Kamel, O. Al Koky and A. Al Dakkak, “Distortion-Free Watermarking Scheme for Wireless Sensor Networks,” in International Conference on Intelligent Networking and Collaborative Systems, Barcelona, Spain, 2009.

[3] B. Wang, X. Sun, Z. Ruan and H. Ren, “Multi-mark- Multiple Watermarking Method for Privacy Data Protection in Wireless Sensor Networks,” Information Technology Journal, vol. 10, no. 4, pp. 833-40, 2011.

[4] I. Kamel and H. Juma, “Simplified watermarking scheme for sensor networks,” International Journal of Internet Protocal Technology, vol. 5, no. 1-2, pp. 101-11, 2010.

[5] U. Khadam, M. Iqbal, M. Alruily, M. Al Ghamdi, M. Ramzan and S. Almotiri, “Text Data Security and Privacy in the Internet of Things: Threats, Challenges, and Future Directions,” Wireless Communications and Mobile Computing, vol. 2020, pp. 1-15, 2020.

[6] R. Wazirali, R. Ahmad, A. Al-Amayreh, M. Al-Madi and A. Khalifeh, “Secure Watermarking Schemes and Their Approaches in the IoT Technology: An Overview,” Electronics (Basel), vol. 10, no. 14, pp. 1744-1772, 2021.

[7] R. Saxena, N. Tiwari and M. K. Ramaiya, “A Survey Work on Digital Watermarking,” International Journal of Latest Technology in Engineering, Management & Applied Science, vol. 4, no. 5, pp. 35-37, 2015.

[8] X. Yu, C. Wang and X. Zhou, “Review on Semi-Fragile Watermarking Algorithms for Content Authentication of Digital Images,” Future Internet, vol. 9, no. 4, pp. 56-73, 2017.

[9] J. Park, S. Jeong and C. Kim, “Robust and Fragile Watermarking Techniques for Documents Using Bi-directional Diagonal Profiles,” Information and Communications Security, vol. 2229, pp. 483-494, 2001.

[10] G. Zhang, L. Kou, L. Zhang, C. Liu, Q. Da and J. Sun, “A New Digital Watermarking Method for Data Integrity Protection in the Perception Layer of IoT,” Security and Communication Networks, vol. 2017, pp. 1-12, 2017.

[11] P. Singh and R. S. Chadha, “A Survey of Digital Watermarking Techniques, Application and Attacks,” International Journal of Engineering and Innovative Technology, vol. 2, no. 9, pp. 165-175, 2013.

[12] Y. Li, H. Guo and S. Jajodia, “Tamper detection and localization for categorical data using fragile watermarks,” in Association for Computing Machinery, New York, 2004.

[13] H. Guo, Y. Li and S. Jajodia, “Chaining watermarks for detecting malicious modifications to streaming data,” Information sciences, vol. 177, no. 1, pp. 281-98, 2007.

[14] I. Kamel and H. Juma, “A lightweight data integrity scheme for sensor networks,” Sensors, vol. 11, no. 4, pp. 4118-36, 2011.

[15] B. Wang, W. Kong, W. Li and N. N. Xiong, “A Dual-Chaining Watermark Scheme for Data Integrity Protection in Internet of Things,” Computer, Materials & Continua, vol. 58, no. 3, pp. 679-95, 2019.