He et al. EURASIP Journal on Image and Video Processing (2020) 2020:12
https://doi.org/10.1186/s13640-020-00500-y

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Semi-structured data protection scheme based on robust watermarking

Jiahuan He[1], Qichao Ying[1], Zhenxing Qian[2*], Guorui Feng[1] and Xinpeng Zhang[2]

Abstract

Semi-structured data is a widely used text format for data interchange and storage. This paper proposes a robust watermarking scheme of data protection for semi-structured data, which uses JSON format as an example for illustration. We first parse JSON file into a data structure of distinct pairs. Afterwards, we generate a transfer matrix to get the intermediate sequences, which are then encoded using error-correction codes and embedded into the pairs. A private key is shared by the data hider and the recipient to resist collusion attacks. On the recipient's side, data extraction can be successfully carried out even the received stego data are tampered. The imperceptibility is realized by embedding data into the less significant digits of numeric data in the cover file. The proposed scheme can be extended on several other formats. The experimental results show that the proposed scheme is robust to various kinds of typical attacks such as contextual truncating, modification, and redundancy addition.

Keywords: Copyright protection, Collusion attack, Semi-structured data, Watermarking

1 Introduction

Semi-structured data are widely used for information storage and data interchange for its convenience and efficiency in data transmission. It is a form of data which contains tags to separate semantic elements and allows hierarchies of records and fields within the data. Some typical applications of semi-structured data are Extensible Markup Language (XML), JavaScript Object Notation (JSON), etc. Nowadays, the dissemination of digital content and data has brought about a series of problems, for example, contextual tampering and illegal duplication. The prevalence of semi-structured data highlights the importance of the protection of data and their copyrights.

Digital watermarking has been used in copyright protection and content authentication for various kinds of data formats. Secret data are embedded into the cover data in an imperceptible way, and can be successfully extracted regardless of attacks during data transmission. Robustness and fidelity are two basic requirements for digital watermarking. The embedded watermark must be robust enough against a variety of possible attacks such as contextual truncating or tampering that can destroy the watermark existence. Meanwhile, the watermarked data need to be close enough to the original data so that users can barely distinguish the differences between them. Generally, there is a trade-off between robustness and fidelity. A stronger robustness usually causes lower fidelity, and vice versa.

In the last decades, researchers mainly focus on the advancement of watermarking technology in multimedia, including algorithms for audio [1–3], image [4–8], and video [9–11]. Researchers have also proposed many watermarking schemes for texts and databases. For text watermarking, a typical method is conducting data embedding based on document structures by shifting rows and columns [12]. In semantic schemes, He et al. [13] use synonym of the substitution and chaotic map to embed the watermark. In syntactic schemes, Mali et al. [14] propose a technique which used grammatical rules like conjunctions and pronouns to generate the watermark. Image-based schemes [15] convert the text into an image at first. For the certain document format, Zhang et al. [16] define new characters using TrueType

* Correspondence: zxqian@fudan.edu.cn
[2]Shanghai Institute of Intelligent Electronics and Systems, School of Computer Science, Fudan University, Shanghai, China
Full list of author information is available at the end of the article

for Microsoft Word to embed the watermark into the Word document. Kuribayashi et al. [17] propose a scheme which used the spaces between words and lines for PDF format. Database watermarking schemes can be classified into two types: distortion-based watermarking and distortion-free watermarking [18, 19]. The distortion-free watermarking uses the data itself to generate watermarks and is generally a fragile watermark. Oppositely, the distortion-based watermarking schemes modify the data with the minimal impact to embed the watermark. The watermarking technology utilizing contextual redundancy [20] embeds the watermark in the redundant bits or meaningless strings. The technology based on data classification [21, 22] is to find the classification data in the database, i.e., the data with a limited value for embedding. Recently, some schemes based on reversible watermarking technology are proposed. Zhang et al. [23] use expansion on data error histogram. Jawad et al. [24] propose a scheme based on reversible and difference expansion, which estimates the distortion using the mean and standard deviation of the watermark relationship.

However, there was little effort paid on copyright protection for the semi-structured data. The abovementioned schemes for text watermarking are generally inapplicable or not effective on semi-structured data. Also, the schemes are generally not robust enough against some typical attacks such as deletion. In this paper, we propose a robust watermarking scheme for semi-structured data protection. For simplicity, we take JSON format as a representative to present the scheme. The cover data is preprocessing ahead of data embedding, which generates valid embedding pairs for the file. We generate pseudorandom sequences using the hashing result of the keys of the pairs as seeds. We then segment the watermark data and construct a transfer matrix to transfer the segments into intermediate sequences. The sequences are further encoded using error-correction coding and embedded into the pairs. A private key is shared by data hider and recipient to resist collusion attacks. On the recipient's side, data extraction can be successfully carried out even the received watermarked data are modified. The experimental results show that the proposed scheme is robust to various kinds of typical attacks. The proposed scheme can be easily extended on other typical formats, e.g., sheet-formatted data like Microsoft Excel or comma-separated values (CSV).

Figure 1 illustrates an application of the proposed scheme. The data hider embeds different watermarks for different users before sharing the cover files. The existence of watermarks is imperceptible and does not affect normal usage of the file. If the recipients illegally upload the data to the Internet, the data hider can locate the source of the information leakage by conducting watermark extraction from the leaked file.

The rest of the paper is organized as follows. In Section 2, we present the embedding and extracting procedures of the proposed scheme. Experimental results are presented in Section 3, and Section 4 concludes the paper.

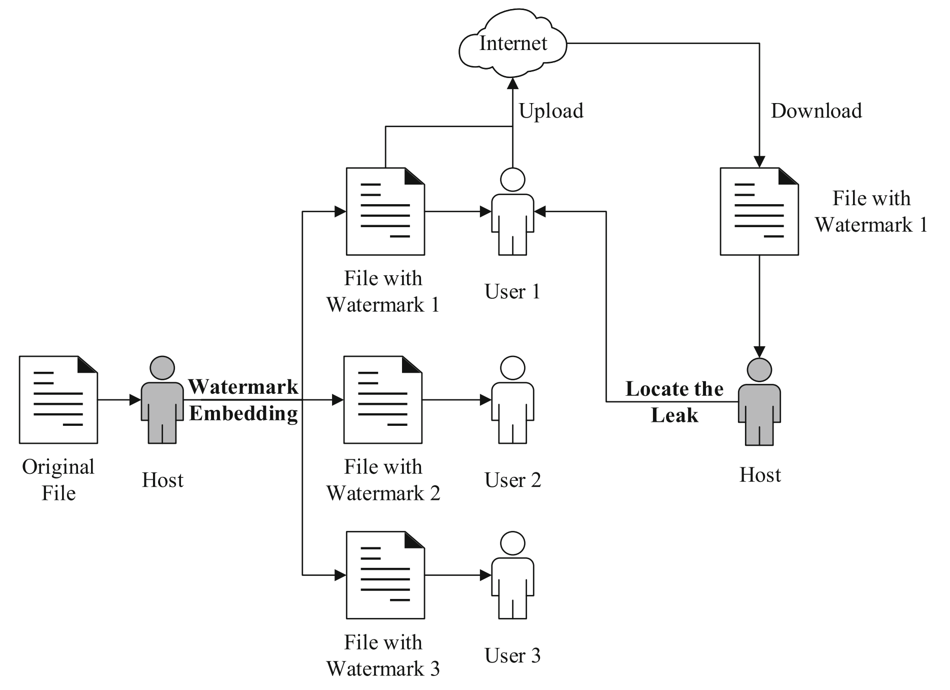


Fig. 1 An application of the proposed scheme

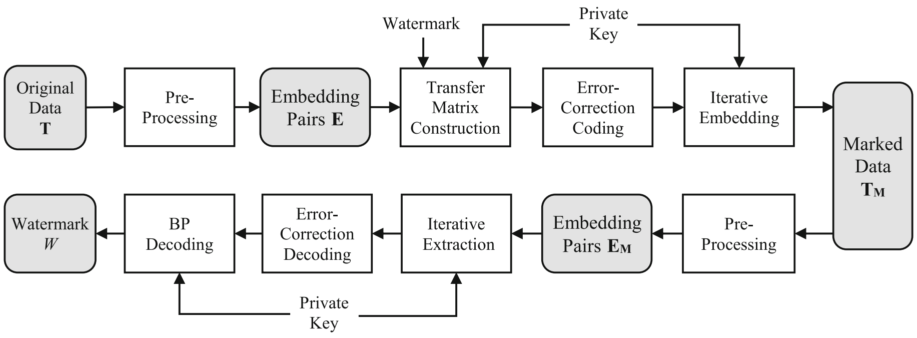


Fig. 2 Overview of the embedding and extraction procedure

2 Proposed method

In this section, we depict the proposed watermarking scheme for semi-structured data. The overview of embedding and extraction procedure is shown in Fig. 2. The right-going flows represent the procedures of watermark embedding, and the left-going flows represent the procedures of watermark extraction. Note that matrices and sets are written in boldface, and sequences are represented in italic.

2.1 Preprocessing

Denote the watermark sequence as W with length l_w. We divide the watermark sequence W into k segments. $W = \{w_1, w_2, w_3, \ldots, w_k\}$. The length of each w_i is denoted as p where $p = l_w/k$.

Figure 3 a provides an example of a cover JSON file. JSON file employs hierarchical or parallel format for data storage. The elements in a JSON file can be categorized into three types, namely, JSON objects that hold data in a hierarchical way, JSON array that holds elements in a parallel way, and JSON primitives that directly hold data. JSON objects and JSON primitives contain pairs of keys and values, while JSON arrays carry parallel data without having their own keys. We use a tree structure to represent the hierarchies of data in JSON, as shown in Fig. 3 b, where the received JSON data is considered the root JSON object, and data are stores in all the leaves.

We parse the cover JSON file, denoted as J, into pairs P, which are the basic units for data embedding in the scheme. Each pair consists of a key and a value. We first initialize P as an empty set, and build the tree structure T for J to get the hierarchical relation. Then we start from T and go through all nodes in the tree. If we meet a JSON object during traversal, we append its key to the end of *prefix*, where *prefix* is originally an empty string. Then we append a "_" in the end to represent the hierarchical relation, and continue to visit the children of this node. If we meet a JSON array, we simply visit the elements in parallel. If we meet a JSON primitive, we append its key to *prefix*. The pair of the corresponding leaf node is p_i=*{prefix, value}*, where *value* is the value of that leaf node. Then we conduct backtracking and continue traversing till all the nodes are visited. After parsing, the pairs $P = \{p_0, \ldots, p_q\}$ that consists of all pairs constructed from leaf nodes of the tree. q denotes the amount of leaf nodes. Fig. 3 c illustrates the parsing result of the file in Fig. 3 a.

To avoid semantic changes, we only hide data into pairs P_{num} whose value only contains numeric data. Other pairs will be excluded during data embedding, whose values will remain unchanged. Also, the data

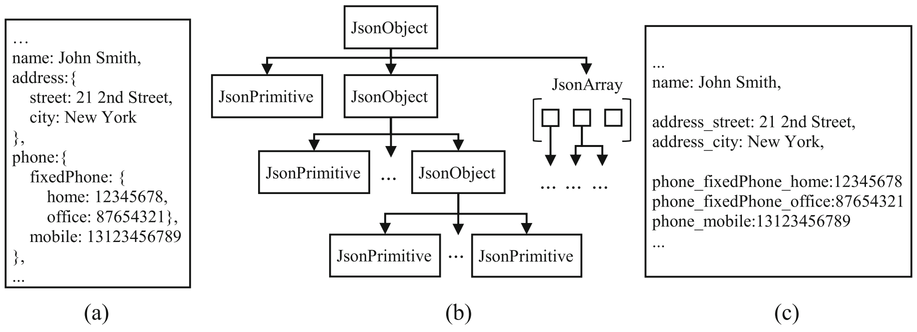


Fig. 3 Diagrams for preprocessing (**a**) cover JSON file, its (**b**) file structure, and (**c**) embedding pairs after parsing

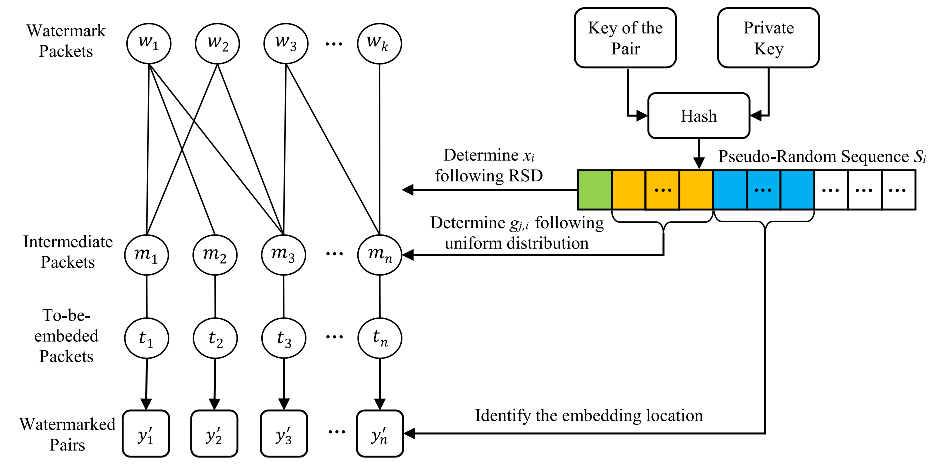

Watermark Packets
w_1 w_2 w_3 ⋯ w_k
Key of the Pair
Private Key
Hash
Pseudo-Random Sequence S_i
Determine x_i following RSD
Determine $g_{j,i}$ following uniform distribution
Intermediate Packets
m_1 m_2 m_3 ⋯ m_n
To-be-embeded Packets
t_1 t_2 t_3 ⋯ t_n
Watermarked Pairs
y'_1 y'_2 y'_3 ⋯ y'_n
Identify the embedding location

Fig. 4 Sketch of watermark embedding procedure

hider can define a set of pairs P_{ban} that are excluded from data embedding by offering their keys. For example, the information of telephone numbers, ids, or timestamps should be excluded since they do not tolerate any modification. Therefore, we only embed data into the rest of the pairs $P' = P_{\text{num}} - P_{\text{ban}}$. We denote the valid pairs as $p'_i \in P', i = \{1, 2, ..., n\}$, where n is the size of P'.

Further, to alleviate embedding distortion, data hider can define a maximal-accepted modifying tolerance T. Denote respectively the value of x'_i and y'_i as $V(x'_i)$ and $V(y'_i)$. $V(y'_i)$ must be within the range of $[(1-T)V(x'_i), (1+T)V(x'_i)]$. We first exclude the leading zeros from the digit sequence. Then we further exclude b leading digits of x'_i to form the embeddable digit sequence

$$b = \lfloor \log(T) \rfloor \tag{1}$$

where $\lfloor x \rfloor$ represents taking the largest integer smaller than x. For example, when the tolerance is set as 0.1, b = 2, so we exclude the leading two digits from the embedding procedure.

Denote the required embedding length as s. We do not embed data into the pair if $\text{len}(V(x'_i)) - b \le s$. Here, $\text{len}(x)$ represents the length of string x. Therefore, the abovementioned pairs are excluded from P' as well. The calculation of s will be discussed in Section 2.2.

We further apply hashing algorithm on the keys of each x'_i so that each pair has a unique hash h'_i. The calculation is as follows.

$$h'_i = \text{Hash}\left(x'_i\right) = \sum_{i=0}^{N-1} \left[\left(x'_i\right)_k \times \sigma^{N-i-1}\right] \tag{2}$$

where $(x'_i)_k$ takes the ASCII code of the k^{th} character of x'_i. σ is a biased parameter, which is set as the private key K shared between the data hider and the recipient. h'_i is ensured to be unique in the cover data.

Afterwards, we generate a set of sequences $S_{\text{rand}} = \{S_1, S_2, ..., S_n\}$, where S_i is pseudorandomly generated by using h'_i as seed, and the elements in S_i are within the range of [0, 1]. Finally, the pairs in P' are sorted in ascending order according to the lexicographical order.

2.2 Watermark embedding (Fig. 4)

We first construct an intermediate sequence M that consisting of n segments. $M = \{m_1, m_2, m_3, ..., m_n\}$. The length of each m_i is also p.

Afterwards, we define a transfer matrix $\boldsymbol{G}$ as follows:

$$\boldsymbol{G} = \begin{bmatrix} g_{1,1} & \cdots & g_{1,n} \\ \vdots & \ddots & \vdots \\ g_{k,1} & \cdots & g_{k,n} \end{bmatrix} \tag{3}$$

where $\boldsymbol{G}$ is a $k \times n$ sparse matrix that each $g_{i,j}$ in $\boldsymbol{G}$ either equals to 0 or 1.

The construction the sparse matrix $\boldsymbol{G}$ is specified as follows. First, we generate the cumulative distribution function cdf_{RSD} of robust soliton distribution. The probability density P_i of RSD is defined as follows:

$$P_i = \frac{\rho_i + \tau_i}{\sum_i (\rho_i + \tau_i)} \tag{4}$$

where ρ_i is the ideal soliton distribution, and τ_i is defined as:

$$\tau_i = \begin{cases} \dfrac{R}{ik}, & 1 \le i \le \lceil k/(R-1) \rceil \\ \dfrac{R\ln\left(\dfrac{R}{\delta}\right)}{k}, & i = \lceil k/R \rceil \\ 0, & \text{others} \end{cases} \tag{5}$$

where $R = c\sqrt{k}\ \ln\left(\frac{k}{\delta}\right)$ denotes the expected ripple size.

For the i^{th} column of $\mathbf{G}$, we flip certain amount of different $g_{j,\ i}$ from zero to one according to the following principle.

$$\begin{cases} x_i = \text{cdf}_{\text{RSD}}(S_i[1]) \\ g_{j,i} = \begin{cases} 0, & \text{Uni}(S_i[k]) \mod n = j \\ 1, & \text{otherwise} \end{cases} \end{cases} \tag{6}$$

where $a[b]$ represents taking the b^{th} element from sequence a, and x_i determines the number of selected $g_{j,\ i}$ that should be flipped. $k \in [S_i[1] + 1, \ldots, S_i[1] + x_i]$. Therefore, the total numbers of ones in each column also follows the robust soliton distribution (RSD).

We further define the procedure of transferring W to M as:

$$M = W \otimes \boldsymbol{G} \tag{7}$$

Here, $\otimes$ represents XOR operation between matrices, in which each $w_i \in W$ is calculated respectively by:

$$m_i = \sum_{j=1}^{k} w_j g_{i,j} \tag{8}$$

Then, the intermediate sequence M can be generated.

Afterwards, we iteratively embed M into the pairs $P^{'}$. For an intermediate packet m_i, we apply cyclic coding as error correction coding. Then the to-be-embedded packet t_i can be generated as $t_i = \text{Cyc}(m_i)$. where the length of t_i is denoted as s ($s > p$). Next, we transfer the data format of $x_i^{'}$ from numbers to string, and use least significant bit (LSB) modification to conduct data embedding. Denote the embedded data of $x_i^{'}$ as $y_i^{'}$.

We then embed s bits to a value of a pair. Denote the value is X and the to-be-embedded bits are a. Then, according to S_i, $a[j]$ will be embedded in $(\Delta + j) \bmod \text{len}(X)$, where $\Delta = S_i[1] + x_i$, and therefore

$$Y[\Delta + j] = X[\Delta + j] - \text{mod}(X[\Delta + j], 2) + a[j] \tag{9}$$

where Y denotes the watermarked value. We repeat this operation until all s bits are embedded into the pair and then iteratively conduct the same embedding procedure to all the pairs in the cover file.

Finally, we get the watermarked pairs as $Y^{'} = \{y_1^{'}, y_2^{'}, \ldots, y_n^{'}\}$. At last, we restore the parsed pairs to the original format. We go through all the nodes of the tree structure. If the values are different from those of $Y^{'}$, we modify them to $y_i^{'}$. So that the watermarked JSON file $\mathbf{J}^{'}$ is obtained.

2.3 Watermark extraction

When the recipient gets the doubted semi-structured data $\mathbf{J}^{*}$, the hidden watermark can be detected and extracted. The flowchart of the extracting procedure is depicted in Fig. 5. Note that the recipient is convinced to know the private keys used in watermark embedding for the generation of random sequences; otherwise, he cannot perform the watermark extraction.

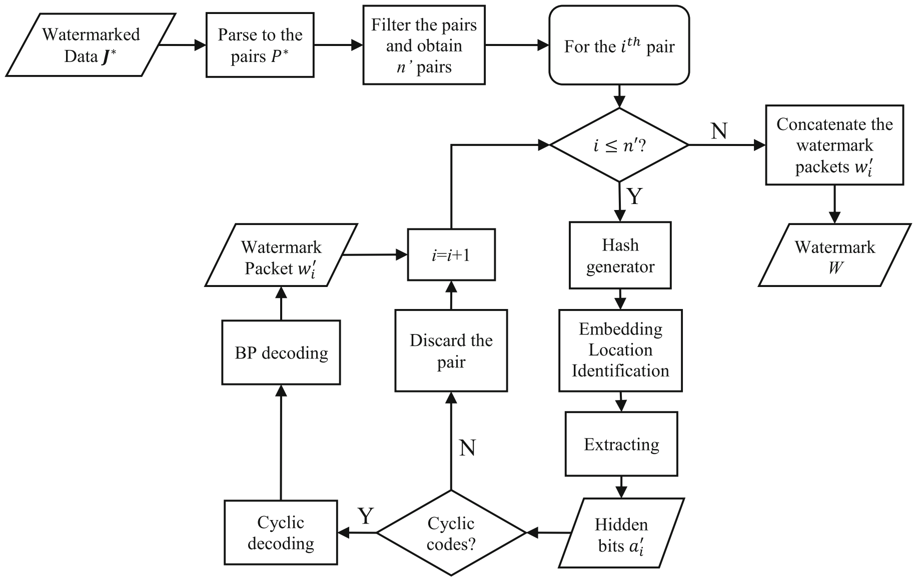


Fig. 5 Flowchart of watermark extraction

The recipient first conducts preprocessing, that parses $\mathbf{J}^*$ into P^*. He applies the same hashing algorithm on the key and generate the pseudorandom sequence S_i, which is the same as the sequence during data embedding. Afterwards, he starts watermark extraction by initializing $\mathbf{G}$ as an empty $n \times k$ matrix. Then, $S_i[1]$ is used to get x_i, and $[S_i[1]+1, \ldots, S_i[1]+x_i]$ are used to flip $g_{j,\ i}$, as introduced in Section 2.2. Then he can get the exact transfer matrix $\mathbf{G}$ used for data embedding.

Afterwards, the pairs in P^* are sorted in ascending order according to the lexicographical order. For each pair, denote the modified value as Y, and the embedded bits as a. According to S_i, $a[j]$ will be embedded in $(\Delta + j) \bmod \mathrm{len}(X)$, where $\Delta = S_i[1] + x_i$, and therefore the recipient gets the hidden bits by (10):

$$a[j] = \bmod(Y[\Delta + j], 2), j = 1, 2, \ldots, s \tag{10}$$

Then he applies cyclic decoding on a to recover the intermediate sequence a_i from $a_i^{'}$. If $a_i^{'}$ is not a valid cyclic code, the recipient is convinced that the data included in the pair is tampered. Thus, he does not utilize the packet for watermark extraction.

For valid packets, he further uses belief propagation (BP) for decoding. He first finds the intermediate packets whose x_i=1, i.e., those which are connected to only one watermark packet. After recovering these packets, they are utilized to decrease the degree of corresponding packets, for example, a packet whose x_i=2 and is connected to the known packet can also be recovered. By iteration, all the source packets can be decoded. Finally, he gets the watermark by concatenating all the recovered packets.

2.4 Extension

The proposed scheme can be extended on sheet-formatted data as well. We take CSV format for an illustration. The embedding and extraction procedures of CSV watermarking using the proposed method can be concluded by Fig. 2 as well. We denote the data contained in the cell located in the ith row and jth column as $c_{i,\ j}$. Also, we denote the watermark sequence as W, and W is equally divided into k segments. $W = \{w_1, w_2, w_3, \ldots, w_k\}$. The length of each w_i is p.

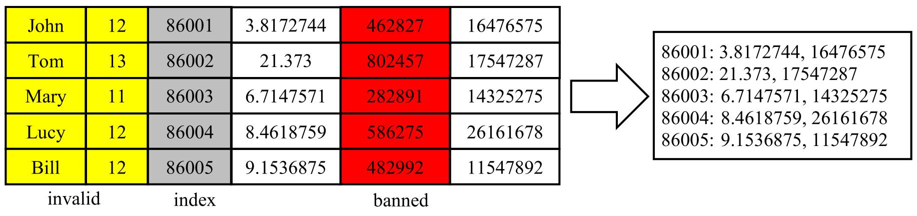
In the preprocessing stage, a cover CSV file is first parsed to the data structure of pairs. Denote the sheet as r rows and l columns, and the rows are regarded as pairs. We first choose a column that satisfies the following two requirements: (1) all the elements inside are not empty and (2) the appearance of repeated elements is the least. The selected column is regarded as index column, denoted as t. For k^{th} row of the file, the key of pair k is $c_{k,\ t}$.For the rest cells in the row, we define the collection $C_k = \{c_{k,v_0}, c_{k,v_1}, \ldots, c_{k,v_l}\}$ that contains valid data for watermark embedding, where cells containing non-numeric data are also excluded to avoid semantic changes. Similarly, the data hider can define a collection of banned columns denoted as C_{ban}, and $C_k \cap C_{\mathrm{ban}} = \varnothing$. Therefore, cells that are not included in C_k will remain unchanged during embedding. To alleviate embedding distortion, b leading digits of $x_i^{'}$ are excluded according to (1) given a threshold T. Here, if the length of c_{k,v_i} is less than b, c_{k,v_i} is excluded from C_k. We further sort the collection C_k according to the length of the values of the cells. Denote the sorted collection as $C_k^{'} = \{c_{k,v_0}^{'}, c_{k,v_1}^{'}, \ldots, c_{k,v_r}^{'}\}$, where r is the remaining number of valid values in k^{th} row, and $len(c_{k,v_0}^{'}) \geq \mathrm{len}(c_{k,v_1}^{'}) \geq \ldots \geq \mathrm{len}(c_{k,v_r}^{'})$. Finally, the *pair* k is generated as P_k=$\{c_{k,\ t}, C_k^{'}\}$, and the valid pairs are denoted as $\mathbf{P} = \{P_0, P_1, \ldots, P_q\}$, where q denotes the amount of valid pairs. Invalid pairs are defined by (11).

$$C_i^{'} = \begin{cases} \text{valid}, \sum_{j=1}^{c} \mathrm{len}\left(c_{i,j} - b\right) \geq s \\ \text{invalid, otherwise} \end{cases} \tag{11}$$

where s is the minimum required length for embedding. Therefore, the invalid rows will be excluded during watermark embedding. The sketch of preprocessing of a cover CSV file is shown in Fig. 6.

We generate hash code for each pair using (2), where $x_i^{'}$ in the equation is replaced with $c_{i,\ t}$, which is the key of packet i. Afterwards, pseudorandom sequences $S_{\mathrm{rand}} = \{S_1, S_2, \ldots, S_r\}$ can be generated under the same principle.

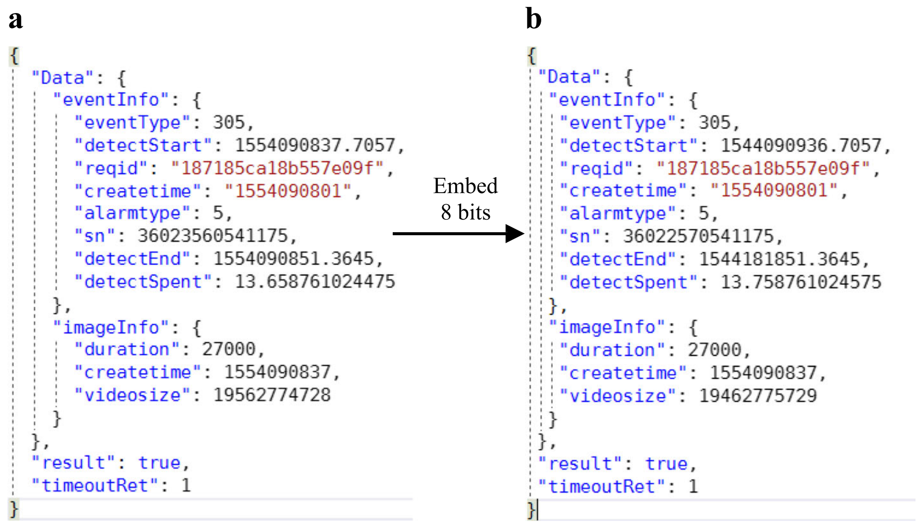
The procedures of transfer matrix construction, error correction encoding is similar to that of semi-structured data. In the iterative embedding stage, for i^{th} row, the to-be-embedded cyclic code is further fragmented and



John	12	86001	3.8172744	462827	16476575
Tom	13	86002	21.373	802457	17547287
Mary	11	86003	6.7147571	282891	14325275
Lucy	12	86004	8.4618759	586275	26161678
Bill	12	86005	9.1536875	482992	11547892

86001: 3.8172744, 16476575
86002: 21.373, 17547287
86003: 6.7147571, 14325275
86004: 8.4618759, 26161678
86005: 9.1536875, 11547892

Fig. 6 Sketch of preprocessing of a cover CSV file



```
{
  "Data": {
    "eventInfo": {
      "eventType": 305,
      "detectStart": 1554090837.7057,
      "reqid": "187185ca18b557e09f",
      "createtime": "1554090801",
      "alarmtype": 5,
      "sn": 36023560541175,
      "detectEnd": 1554090851.3645,
      "detectSpent": 13.658761024475
    },
    "imageInfo": {
      "duration": 27000,
      "createtime": 1554090837,
      "videosize": 19562774728
    }
  },
  "result": true,
  "timeoutRet": 1
}
```

```
{
  "Data": {
    "eventInfo": {
      "eventType": 305,
      "detectStart": 1544090936.7057,
      "reqid": "187185ca18b557e09f",
      "createtime": "1554090801",
      "alarmtype": 5,
      "sn": 36022570541175,
      "detectEnd": 1544181851.3645,
      "detectSpent": 13.758761024575
    },
    "imageInfo": {
      "duration": 27000,
      "createtime": 1554090837,
      "videosize": 19462775729
    }
  },
  "result": true,
  "timeoutRet": 1
}
```

Fig. 7 Embedding test for a JSON file. **a** Cover data. **b** Stego data

embedded into C_i'. For c_{i,v_0}', we embed $\lfloor \text{len}(c_{i,v_0}')/s \rfloor$ bits into the value. We iteratively embed all the bits into C_i' until data embedding is finished. Thus, cells with longer value are more likely to carry more additional bits.

On the recipient's side, data preprocessing for sheets is also carried out ahead of watermark extraction. The recipient gets the pairs and the corresponding pseudorandom sequences. Then he can successfully construct the transfer matrix. Under the same principle, he knows how many bits are hidden in a given $c_{i,j}'$, and he extracts the secret bits with the help of the retrieved pseudorandom sequence. He then checks whether the recovered data sequence is a valid cyclic code. Finally, back propagation is also applied for data extraction.

3 Results and discussion

To verify the proposed scheme, we have conducted many experiments on an amount of JSON and CSV files. The sources are provided by some companies that are allowed for experimental uses. We use binary random sequences as digital watermark. We test the watermark robustness to several attacks.

Figure 7 shows two short JSON files before and after embedding 8 bits watermark as an example. From Fig. 7, we can easily observe that the long floating numbers are modified to carry the secret data, while the shorter ones and strings remain unchanged. Thus, we can consider the modification is imperceptible.

3.1 Settings and evaluations

The main parameters in the proposed scheme are the length of each watermark packet p, the length of to-be-embedded packet s, and two parameters of RSD c and δ as is described in Section 2. In the experiment, we set $p = 4$, $s = 7$, $c = 0.1$, $\delta = 0.5$.

The proposed scheme is low in computational complexity, which makes it easier to be embedded on hardwares. The watermark embedding and extraction can be done within several seconds by a personal laptop with 2.60 GHz CPU and 8.00 GB RAM.

For an objective performance assessment of the proposed scheme, the overall distortion after watermark embedding can be defined as:

$$D = \sum_{i=1}^{N} \frac{U_i A_i}{N} \tag{12}$$

where the semi-structured data file contains N key-value pairs, and U_i equals to 0 or 1 indicates whether the i^{th} key-value pair has been modified. A_i represents the modified magnitude. $A_i = |y_i - x_i|/x_i$, where x_i and y_i respectively denote the original value and the modified value. Specially, if the i^{th} key-value pair is deleted, A_i is set to 1. A larger D indicates that the introduced distortion is larger.

3.2 Embedding performances

In Table 1, we measure the introduced distortions according to (12) under different embedding capacities. As is shown, the introduced distortion gradually remains stable even the embedding capacity grows higher, which is consistent with the analysis of our method. Meanwhile, the distortion of data is quite small, which

Table 1 Capacity vs. distortion

Capacity (bits)	8	20	40	60	80
D (%)	0.1078	0.0981	0.1139	0.1069	0.1191

Table 2 Ratios of successful extraction under different attacks for semi-structured data

Type of attack		Ratios of successful extraction			
		File1.json (size: 100)	File2.json (size: 200)	File3.json (size: 800)	File4.json (size: 1500)
No. attack		10/10	10/10	10/10	10/10
Deletion	P_d = 5%	10/10	10/10	10/10	10/10
	P_d = 10%	10/10	10/10	10/10	10/10
	P_d = 20%	10/10	8/10	10/10	10/10
	P_d = 30%	10/10	8/10	9/10	10/10
Value modification	P_m = 5%	10/10	10/10	10/10	10/10
	P_m = 10%	10/10	10/10	10/10	10/10
	P_m = 20%	10/10	10/10	10/10	10/10
	P_m = 30%	10/10	9/10	10/10	10/10
Insertion	P_i = 5%	10/10	10/10	10/10	10/10
	P_i = 10%	10/10	10/10	10/10	10/10
	P_i = 15%	10/10	10/10	10/10	10/10
	P_i = 20%	10/10	10/10	10/10	10/10

indicates the watermarking scheme that has little impact on the original data.

We further test the robustness of semi-structured data watermarking to three types of attack when l_w = 64 bits. Here, typical attacks including pair deletion, value modification, and redundancy insertion are applied in various degrees. We use four JSON files of different sizes for testing: file1, file2, file3, and file4, each containing 100, 200, 800, and 1500 pairs. We perform the same attack for multiple times on each watermarked file. The ratios of successful extraction under different attacks for semi-structured data are shown in Table 2, where extractions with wrongly retrieved bits are not considered successful extraction. In the table, P_d, P_i, and P_m respectively denote the percentage of key-value pairs which are deleted, inserted, or tampered in the file.

The proposed watermark scheme is generally robust to all the abovementioned attacks, even the attacks are strong. Especially, the scheme shows promising embedding performance against pair deletion and redundancy insertion. Also, files with more valid pairs gradually show stronger robustness. The main reason of high robustness is that: for pair deletion, the watermark can be extracted using the remaining valid pairs. For contextual modification, the recipient can identify the tampered location using cyclic code checks and discard the tampered pair. For insertion, data extracted from the inserted pairs can hardly pass the cyclic code checks, and thus they are also discarded.

The proposed scheme is also robust to combined attacks, as is shown in Table 3, e.g., the doubted file is a both modified and truncated version of watermarked file. The results prove that the scheme is credible and applicable in real uses.

We also test the robustness of the extended part of the proposed scheme. We also use several cover CSV files for copyright watermarking. Here, we use the same embedding capacity l_w = 64 bits. The attacks include sorting, row deletion, value modification, row insertion, and column insertion. The test is applied on four CSV files. In Table 4, P_{dr} and P_{ir} are the percentage of rows which are added or inserted to the file. P_m denotes the percentage of values in the sheet which are tampered and l_i denotes the number of inserted columns.

As is shown in Table 4, the extended part of the proposed scheme also shows great robustness to typical attacks. Similarly, CSV files with more rows are stronger in robustness. As for column insertion, if the inserted values are participates in the payload allocation in each row, the extracted bits are distorted and cannot pass cyclic code checks. The robustness against column insertion is restrained in the proposed scheme.

Table 3 Ratios of successful extraction under combined attacks for semi-structured data

Type of attack	Ratios of successful extraction			
	File1.json	File2.json	File3.json	File4.json
Deletion and modification (P_d = 10%, P_m = 10%)	10/10	10/10	10/10	10/10
Deletion and insertion (P_d = 10%, P_i = 10%)	10/10	10/10	10/10	10/10
Modification and insertion (P_m = 10%, P_i = 10%)	10/10	10/10	10/10	10/10

Table 4 Ratios of successful extraction under different attacks for sheets

Type of attack		Ratios of successful extraction			
		File1.csv (size: 500×23)	File2 .csv (size: 1000×6)	File3 .csv (size: 2000×2)	File4 .csv (size: 5000×38)
No Attack		10/10	10/10	10/10	10/10
Sorting		10/10	10/10	10/10	10/10
Row deletion	P_{dr} = 20%	10/10	10/10	10/10	10/10
	P_{dr} = 40%	10/10	10/10	10/10	10/10
	P_{dr} = 60%	10/10	10/10	10/10	10/10
	P_{dr} = 80%	10/10	10/10	10/10	10/10
Value modification	P_m = 20%	10/10	10/10	10/10	10/10
	P_m = 40%	10/10	10/10	10/10	10/10
	P_m = 60%	10/10	10/10	10/10	10/10
	P_m = 80%	10/10	10/10	10/10	10/10
Row insertion	P_{ir} = 10%	10/10	10/10	10/10	10/10
	P_{ir} = 20%	10/10	10/10	10/10	10/10
Column insertion	l_i = 1	10/10	5/10	4/10	10/10
	l_i = 5	9/10	1/10	0/10	8/10

Finally, for the security of the proposed scheme, we use a private key shared by data hider and recipient. Different private key results in different embedding locations, which helps to resist the collusion attack, i.e., two recipients cannot infer the embedding location by comparing two the same files with different watermarks.

4 Conclusions

In this paper, a novel watermarking scheme for semi-structured data protection is proposed. The cover file is firstly parsed into pairs. We generate a transfer matrix to get the intermediate sequences, which are then encoded using error-correction codes and embedded into the pairs. On the recipient's side, data extraction can be successfully carried out even the received data are tampered. The proposed scheme can be extended on several other formats. The experimental results show that the proposed scheme is robust to various kinds of typical attacks such as contextual truncating, modification, and redundancy addition. Meanwhile, the introduced distortion is comparatively low. Finally, a private key also helps to resist collusion attacks.

Abbreviations

XML: Extensible markup language; JSON: JavaScript object notation; SS: Spread spectrum; DFT: Discrete Fourier transform; DCT: Discrete cosine transform; DWT: Discrete wavelet transform; CSV: Comma-separated values; RSD: Robust soliton distribution; BP: Belief propagation

Acknowledgements

Many thanks to the anonymous reviewers for their constructive suggestions to help improving this paper.

Authors' contributions

Our contributions in this paper were that the first author (JH) participated in the designing of the scheme and drafted the manuscript. The second author (QY) carried out the experiments and participated in the designing of the scheme. The third author (ZQ) conceived of the study, participated in the design, and helped to draft the manuscript. The fourth author (GF) and the fifth author (XZ) helped to design and improve the scheme. All authors read and approved the final manuscript.

Authors' information

Jiahuan He received the B.S. degree from Shanghai University, China, in 2018, where he is currently pursuing the M.S. degree. His research interests include image processing and multimedia security. Qichao Ying received the B.S. degree from Shanghai University, China, in 2017, where he is currently pursuing the M.S. degree. His research interests include information hiding, image processing, and multimedia security.
Zhenxing Qian received the B.S. and Ph.D. degrees from the University of Science and Technology of China (USTC), in 2003 and 2007, respectively. He is currently a Professor with the School of Computer Science, Fudan University. He has published over 100 peer-reviewed papers on international journals and conferences. His research interests include information hiding, image processing, and multimedia security.
Guorui Feng received the B.S. and M.S. degree in computational mathematic from Jilin University, China, in 1998 and 2001 respectively. He received Ph.D. degree in electronic engineering from Shanghai Jiaotong University, China, 2005. From January 2006 to December 2006, he was an assistant professor in East China Normal University, China. During 2007, he was a research fellow in Nanyang Technological University, Singapore. Now he is with the school of communication and information engineering, Shanghai University, China. His current research interests include image processing, image analysis, and computational intelligence.
Xinpeng Zhang received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.E. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively, where he has been with the faculty of the School of Communication and Information Engineering, since 2004, and is currently a Professor. His research interests include information hiding, image processing, and digital forensics. He has published over 200 papers in these areas.

Funding

This work was supported by the Natural Science Foundation of China (Grant Grant U1736213, U1636206, and U1936214).

Availability of data and materials
The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Ethics approval and consent to participate
Ain-Shams University Hospitals ethics committee approval; (FMASU R59/2018). Informed consent to participate in the study was taken from all participants.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Communication and Information Engineering, Shanghai University, Shanghai, China. [2]Shanghai Institute of Intelligent Electronics and Systems, School of Computer Science, Fudan University, Shanghai, China.

Received: 1 November 2019 Accepted: 9 March 2020
Published online: 16 March 2020

References
1. Y. Xiang, I. Natgunanathan, D. Peng, et al., Spread spectrum audio watermarking using multiple orthogonal PN sequences and variable embedding strengths and polarities. IEEE/ACM Trans. Audio, Speech, Language Process **26**(3), 529–539 (2018)
2. Z. Liu, Y. Huang, J. Huang, Patchwork-based audio watermarking robust against de-synchronization and recapturing attacks. IEEE Trans. Inf. Forensics Security. **14**(5), 1171–1180 (2019)
3. G. Hua, J. Goh, V.L.L. Thing, Cepstral analysis for the application of echo-based audio watermark detection. IEEE Trans. Inf. Forensics Security. **10**(9), 1850–1861 (2015)
4. T. Zong, Y. Xiang, I. Natgunanathan, et al., Robust histogram shape-based method for image watermarking. IEEE Trans. Circuits Syst. Video Technol. **25**(15), 717–729 (2015)
5. M. Urvoy, D. Goudia, F. Autrusseau, Perceptual DFT watermarking with improved detection and robustness to geometrical distortions. IEEE Trans. Inf. Forensics Security. **9**(7), 1108–1119 (2014)
6. S.-W. Byun, H.-S. Son, S.-P. Lee, Fast and robust watermarking method based on DCT specific location. IEEE Access. **7**, 100706–100718 (2019)
7. D. Zheng, S. Wang, J. Zhao, RST invariant image watermarking algorithm with mathematical modeling and analysis of the watermarking processes. IEEE Trans. Image Process. **18**(5), 1055–1068 (2009)
8. P.-C. Su, Y.-C. Chang, C.-Y. Wu, Geometrically resilient digital image watermarking by using interest point extraction and extended pilot signals. IEEE Tran. Inf. Forensics Security. **8**(12), 1897–1908 (2013)
9. T. Dutta, H. Gupta, A robust watermarking framework for high efficiency video coding (HEVC)—encoded video with blind extraction process. J. Vis. Commun. Image Represent. **38**, 29–44 (2016)
10. J.-S. Tsai, W.-B. Huang, Y.-H. Kuo, On the selection of optimal feature region set for robust digital image watermarking. IEEE Trans. Image Process. **20**(3), 735–743 (2011)
11. M. Amini, M. Ahmad, M. Swamy, A robust multibit multiplicative watermark decoder using vector-based hidden Markov model in wavelet domain. IEEE Trans. Circuits Syst. Video Technol. **28**(2), 402–413 (2018)
12. J. Brassil, S. Low, N. Maxemchuk, et al., Electronic marking and identification techniques to discourage document copying. IEEE J. Sel. Areas Commun. **13**(8), 1495–1504 (1995)
13. L. He, X.-l. Gui, An active attack on chaotic based text zero-watermarking. IEEE Conf. Anthol. 1-4 (2013)
14. M. L. Mali, N. N. Patil, and J. B. Patil, "Implementation of text watermarking technique using natural language watermarks," Proc. Int. Conf. Commun. Syst. Netw. Tchnol. 482-486 (2013)
15. D. Huang, H. Yan, Interword distance changes represented by sine waves for watermarking text images. IEEE Trans. Circuits Syst. Video Technol. **11**(12), 1237–1245 (2001)
16. S. Zhang, Z. Yao, X. Meng, et al., New digital text watermarking algorithm based on new-defined characters, Proc. IEEE Int. Symp. Comput. Consum. Control. 713-716 (2014)
17. M. Kuribayshi, T. Fukushima, N. Funabiki, Robust and secure data hiding for PDF text document. IEICE Trans. Inf. Syst. **102**(1), 41–47 (2019)
18. I. Kamel, A schema for protecting the integrity of databases. Comput. Secur. **28**(7), 698–709 (2009)
19. A. Khan, S.A. Husian, A fragile zero watermarking scheme to detect and characterize malicious modifications in database relations. Sci. World J. **2013**(796726), 1–16 (2013)
20. R. Agrawal, P.J. Haas, J. Kiernan, Watermarking relational data: framework, algorithms and analysis. The VLDB J. **12**(2), 157–169 (2003)
21. R. Sion, Proving ownership over categorical data, Proc. IEEE Int. Conf. Data Eng. (2004)
22. R. Sion, M. Atallah, S. Prabhakar, Rights protection for categorical data. IEEE Trans. Knowl. Data Eng. **17**(7), 912–926 (2005)
23. Y. Zhang, B. Yang, X. Niu, Reversible watermarking for relational database authentication. J. Comput. **17**(2), 59–66 (2006)
24. K. Jawad, A. Khan, Genetic algorithm and difference expansion based reversible watermarking for relational databases. J. Syst. Softw. **86**(11), 2742–2753 (2013)