Systems Enabling the Use of Natural Language Processing Methods for Software-Based Leak Detection, Prevention, and Mitigation

Jacob R. Fuehne
NLP Researcher
University of Illinois
jfuehne2@illinois.edu

Vivek C. Nair
Authentication Researcher
Solid Security
vcnair@solidsecurity.co

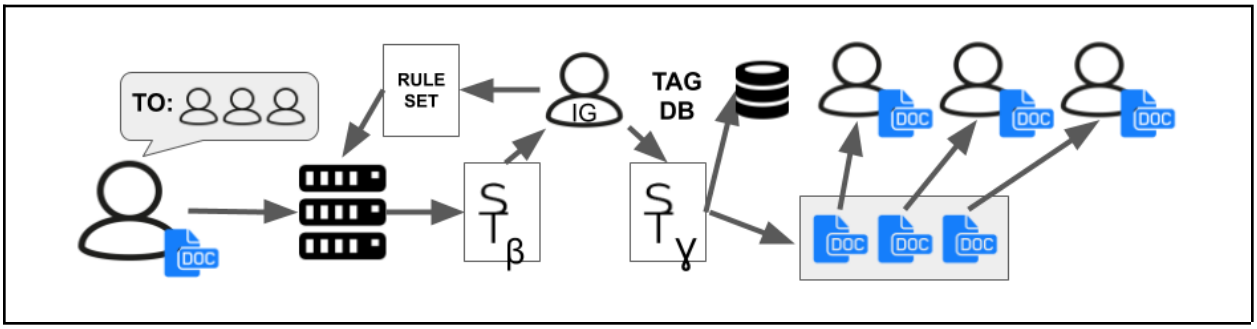

Figure 0: Simplified overview of the proposed system.

Abstract

In the use of software for the detection, prevention, and mitigation of data leaks of primarily textual documents, a method is necessary to make the contents of such documents uniquely identifiable without sacrificing their semantic meaning. We propose a system whereby source documents are transformed, in accordance with a ruleset separately generated per natural language processing (NLP) principles, into a series of permuted documents such that each intended recipient receives a document whose text content is uniquely identifiable but semantically identical. Our proposed system consists of: a document author, providing a source text and a list of intended recipients; a spintaxer, which, using a provided ruleset and source text, generates a beta spintax and statistics thereof; an inspector, providing a ruleset by which the source text shall be transformed and reviewing the provided beta spintax and statistics to generate a gamma spintax; a spinner, which, using a provided gamma spintax and list of recipients, generates uniquely identifiable documents and document identifiers for storage in a database; a tag database, which stores document identifiers along with the recipient of the corresponding document permutation; a spintax database, which stores gamma spintaxes; and an investigator, which determines the source of leaked documents. We additionally propose an advanced "spintax" standard for intermediate storage of acceptable document permutations and unique identification of each allowable permutation.

1. Introduction

1.1. Problem Statement

The task at hand is preventing leaks (i.e. unwanted external disclosures) of sensitive materials, particularly in instances where such disclosures could cause serious damage to the party seeking to prevent disclosure. An additional focus is the issue of detecting when such leaks occur and mitigating the damage caused by such leaks. Necessary for the tasks of prevention, detection, and mitigation of leaks is a method for identifying the source of such leaks after they occur based on the leaked information alone. Providing such a method for text documents is the chief goal of this paper.

Cause of Leak		Applicable?
Personnel Compromise	Intentional	✓
	Unintentional	✓
System Compromise	Localized	✓
	Centralized	✗

Table 1: Applicability of the Tailspin method to the investigation of various leak scenarios.

1.2. Causes of Document Leaks

1.2.1. Personnel Compromise

1.2.1.1. Intentional

The case where a trusted individual or individuals intentionally leak a document, (for example to a press outlet, “watchdog” organization, or competitor/adversary), is chiefly at issue in this paper. Methods that enable unlawfully leaked documents to be traced back to a particular suspect are especially useful in this case because determining the offender and removing their access to sensitive materials is usually sufficient to prevent further leaks from occurring. Thus handling the case of a person intentionally leaking documents, with full knowledge that such actions are unlawful, is our primary motivation in this paper.

1.2.1.2. Unintentional

In cases where a person unintentionally leaks documents (eg. by falling victim to a phishing scheme or other social engineering attack), our method may also be useful for preventing further leaks. If a particular individual is the repeated source of unintentional leaks, the methods outlined herein for identifying the persons responsible for such leaks are vital for identifying and ultimately remedying their root cause. The proposed software-based methods are particularly important when leaks are unintentional because traditional human intelligence (HUMINT) may be unable to identify the sources of such leaks when the persons ultimately responsible are unaware that they are at fault.

1.2.2. System Compromise

1.2.2.2. Localized

Considering, now, the case where a system is compromised, but where that compromise is localized to a small number of users (eg. via the presence of malware on their personal machines or use of insecure passwords amongst those particular users), our method may be useful for preventing further leaks. We can consider this case equivalent to the case where individuals are unintentionally leaking data because the vulnerability exists at a personal scope whether or not those persons are ultimately at fault. Therefore, as before, the methods outlined herein for identifying the persons responsible for such leaks are useful for determining and ultimately remedying their root cause, especially when compared to HUMINT methods of leak prevention.

1.2.2.1. Centralized

Turning finally to the case where a system is compromised at a centralized location, such that the data accessible from compromised systems is common to many if not all members of the organization. The methods proposed in this paper are not adequate to deal with such a case, because the methods proposed to narrow mitigation efforts to a particular user are neither meaningful nor helpful when a breach is not localized to any particular user. However, insofar as using our methods results in documents received by users differing in content from documents stored centrally, determining that a

leaked document is not associated with any particular user may be helpful in concluding that the breach occurred in a centralized location.

1.3. Means of Identification

1.3.1. Metadata

A historical means of allowing documents to be uniquely identifiable has been the addition of unique metadata to the document file. However, these methods are inadequate at protecting the document's content because they protect the file which contains the sensitive data rather than the data itself. Trivial methods, such as copying and pasting the data (if text-based) into another document or manually re-typing the data into another document, may be sufficient to circumvent methods that rely solely on metadata for leak identification. Other methods, such as reformatting the data or photographing the original document, are all plausible means by which the document's sensitive contents may be leaked without its corresponding metadata, thus rendering the leak impossible to trace back to a human source. In general, methods that rely on protecting the container in which secret contents are enclosed, namely the file in which they are stored, rather than protecting the file's secret contents directly, will be vulnerable to this class of attack.

1.3.2. Content

To address the shortcomings of using metadata to identify leaked documents, the remainder of our paper examines uniquely modifying the sensitive contents of a document directly. A key challenge is doing so in a way that is undetectable and does not change the meaning or interpretation of the information contained in the document. This challenge is more easily overcome in some types of documents than others.

1.4. Types of Leaked Documents

1.4.1. Media, Code/Executables

When considering the various leak prevention, detection, and mitigation mechanisms in place for various types of documents, we consider first the case of documents whose primary contents are non-textual. Well-known steganographic techniques exist which allow for the invisible placement of data within images, audio files, video files, executable code, and various other file types. These methods are enabled by the fact that small changes to media files are essentially undetectable to end-users. For example, small, systematic changes to individual pixel values in photos and videos are virtually indistinguishable to the naked eye. Extant cryptographic methods employ these small modifications to encode data in compatible media formats such that they can ultimately be traced back to an individual user. Similar techniques exist for source code, executable files, raw data, and various other file types that are not primarily textual. Although the focus of this paper is not on these methods, they can be effectively combined with the methods outlined herein for mixed-media formats (such as text documents with embedded images) for comprehensive coverage.

1.4.2. Text

We consider next the case of documents that are primarily textual, and where any media contained within exists mainly to complement or illustrate the textual content rather than being the primary unit of transferring information. Steganography is difficult to apply in these cases because small, random changes to document text become immediately apparent to end recipients. Systematically changing document contents such that the text remains syntactically and grammatically correct and retains its semantic meaning is the primary aim of this paper.

Distribution Model	File is identifiable?	Text is identifiable?
Direct	✗	✗
Metadata Tagging	✓	✗
Tailspin	✓	✓

Table 2: Traceability characteristics of various document distribution models.

2. Document Distribution Models

2.1. Trivial (Direct) Model

In the course of examining the different models which may be employed to communicate the contents of a source document to a set of end recipients, it is useful to first examine the trivial model where

documents are distributed directly to intended recipients unaltered (Fig. 1). This model provides no meaningful way to determine the source of a leaked document; when a document is leaked, it is likely to be identical to the version received by every document recipient.

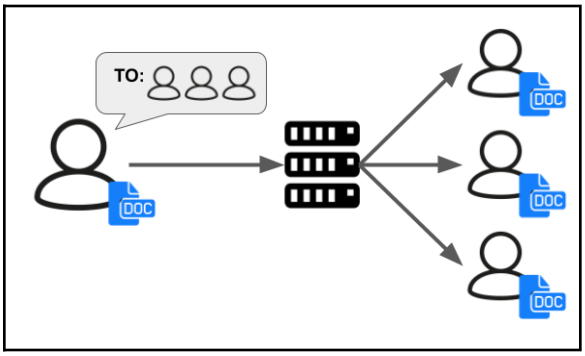

Figure 1: Trivial document distribution system.

A distribution server is often used to deliver the document from its author to a list of intended recipients provided by the author. This server may perform additional security checks before delivering the document (eg. confirming that all intended recipients have the necessary security clearance to receive the document).

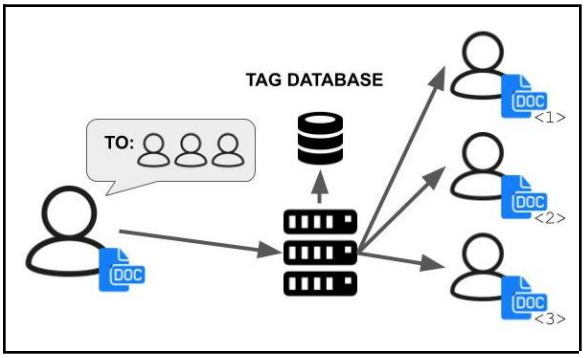

Figure 2: Document distribution system employing metadata-tagging.

2.2. Metadata-tagging Model

In the metadata-tagging model of document distribution (Fig. 2), a document is sent by its author to a distribution server along with a set of intended recipients. The distribution server then embeds a unique piece of metadata within each document prior to delivery to each recipient. The metadata associated with each recipient is stored in a tag database for later reference.

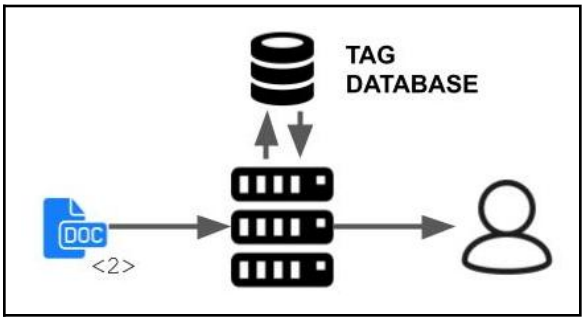

Figure 3: Post-leak analysis mechanism in metadata-tagging paradigm.

In the event of a breach, the source can be identified by extracting the stored metadata from the leaked document and comparing it with the records present in the tag database to determine the responsible user (Fig. 3). However, as discussed previously, this method merely protects the file in which the sensitive material is stored rather than the sensitive material itself, and as such can be trivially bypassed.

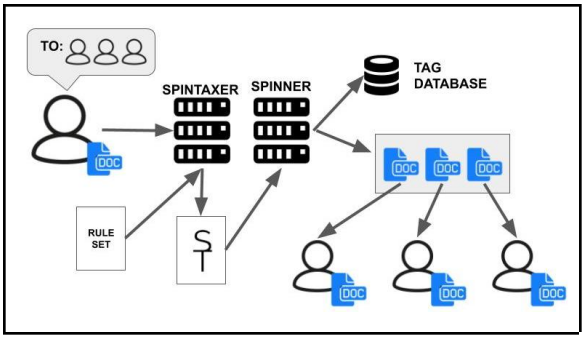

Figure 4: Simple Tailspin document generation and distribution system.

2.3. "Tailspin" Model

Our proposed model, "Tailspin," aims to permute an original document into several documents with different text content for each recipient such that the documents are uniquely identifiable based on their content alone (Fig. 4). In the first stage, a "spintaxer" uses a source document and a provided ruleset to generate a "spintax," an intermediate document describing all of the ways in which the document may reasonably be permuted. In the second stage, a "spinner" uses the generated spintax to produce a

series of permuted documents, each associated with a unique "tag" or identifier. Unique documents are then delivered to each intended recipient, and records are maintained in the tag database to associate each document permutation with its corresponding recipient. The spintax used to generate the documents is stored securely for later reference.

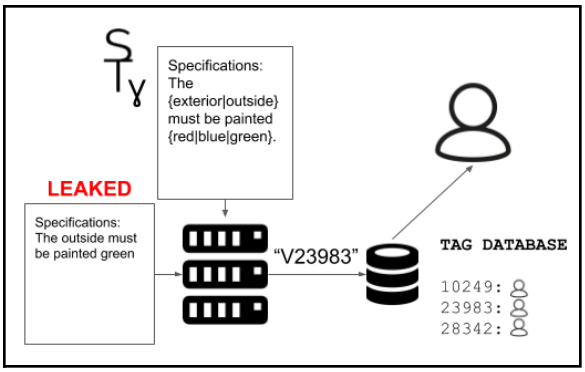

Figure 5: Post-leak analysis mechanism in Tailspin paradigm.

In the event of a breach, the leaked document can be combined with the stored spintax to recover the tag associated with that particular document permutation (Fig. 5). That tag can then be checked against the tag database to determine which user was the source of the breach. The specialized design of an advanced spintax format in which each possible permutation is uniquely indexed and this index is algorithmically determinable from the permuted document is a critical component of this mechanism.

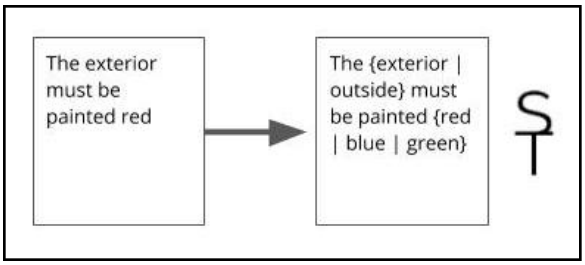

Figure 6: The spintaxing process converts an original text into a spintax.

3. Spintax

3.1. Standard Spintax

Spintax (sentence permutation syntax) is an important component of the Tailspin system, acting as an intermediary format specifying valid permutations of a source document. Standard spintax is a widely used format whereby "switches" are used to denote each individual instance where one wishes to specify multiple acceptable variations of a source text. Text that is to be randomized will be surrounded by curly brackets ({}) and each of the possible variations to be generated will be separated by vertical bar characters (|).

An example of the "spintaxing" process, whereby a source text is transformed into a standard spintax, is illustrated in Figure 6. The switches included in the above example, "{exterior|outside}" in the first instance and "{red|blue|green}" in the second instance, indicate that "exterior" and "outside" are both acceptable permutations in the first switch, and "red," "blue," and "green" are all acceptable permutations in the second switch. Figure 7 displays the resulting sentence permutations that are accepted by the spintax.

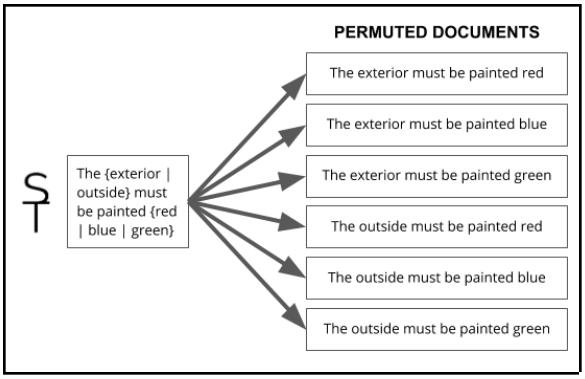

Figure 7: A spintax is transformed into a set of acceptable permuted documents.

One of the permutations generated above will be chosen at random to present to each recipient. In the event of a leak, an investigator could correlate the leaked document to the individual receiving that permutation, that individual then being the prime suspect for leaking the document.

3.2. Solid Spintax

3.2.1. Overview

Many features necessary for representing the full range of permutative options present in complex natural language models are not available in standard spintax. We have developed a new standard, "Solid Spintax," to handle these cases. The intent of the new

standard is to expand the possibilities of standard spintax and to maximize one's ability to generate permutations while maintaining semantic meaning. Solid Spintax implements several new features such as integer switches, nested switches, and global switches.

3.2.2. Integer Switches

Integer switches allow one to decide a range of numbers that are acceptable, from amongst which an integer will be randomly chosen for the final output. The syntax for an integer switch designates that the switch will be surrounded by curly braces ({}) and an inclusive upper bound and lower bound will be separated by a hyphen (-).

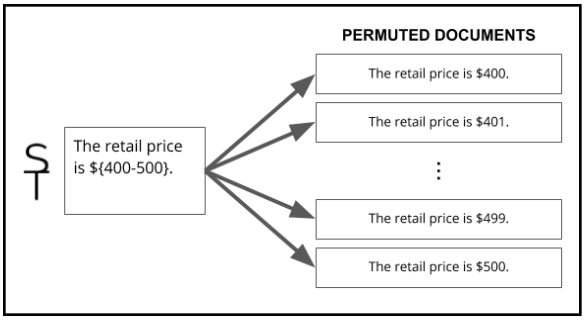

Figure 8: An integer switch in the Solid Spintax standard and corresponding documents.

Figure 8 illustrates the example spintax "The retail price is ${400-500}." This spintax produces 101 possible variations, such as "The retail price is $479." and "The retail price is $434." Large integer ranges contain far more permutations than can be derived from a string switch, so Solid Spintax's support of integer switches dramatically increases document entropy.

3.2.3. Nested Switches

Nested switches are created by specifying a switch within the body of another switch. Solid Spintax supports a theoretically unlimited nesting depth of switches, which facilitates exponential growth in the number of allowable permutations, subject, of course, to hardware limitations and the constraint to retain semantic meaning.

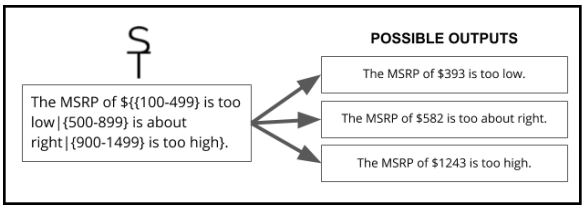

Figure 9: Nested switches example.

Figure 9 illustrates an example where a series of integer switches are nested within a string switch to provide a meaningful output where existing spintax standards would otherwise lose meaning. The spintax "The MSRP of ${{100-499} is too low|{500-899} is about right|{900-1499} is too high}." produces outputs like "The MSRP of $393 is too low," "The MSRP of $582 is too about right," or "The MSRP of $1243 is too high."

Consider the alternative of using parallel switches to represent this option set: "The MSRP of ${100-499} is {too low|about right|too high}." This spintax has the potential of producing confusing and incorrect outputs like "The MSRP of $1495 is too low." or "The MSRP of $100 is too high." This is a perfect example of the importance of nested switches for retaining semantic meaning.

3.2.4 Global Switches

Global switches enable one to apply spintax on words that must maintain uniformity to avoid drawing suspicion to the inconsistencies in the text. A global switch is specified by including an at-sign (@) followed by a switch statement. Global switches must contain the same options globally so as to guarantee the same output as per the current specifications.

```
Introducing project @{Purple Phoenix|Golden Eagle}. Project @{Purple Phoenix|Golden Eagle} is a new security initiative from Solid Security. Our goal with project @{Purple Phoenix|Golden Eagle} is to...
```

Figure 10: Spintax with global switches.

Figure 10 illustrates a spintax with global switches. Upon encountering the first switch, the spinner will choose either "Purple Phoenix" or "Golden Eagle." The choice will be stored, and upon future instances,

the same choice will be made to ensure global consistency. Figures 11 and 12 display the only two allowable output permutations.

Introducing project Purple Phoenix. Project Purple Phoenix is a new security initiative from Solid Security. Our goal with project Purple Phoenix is to...

Figure 11: Spintax with global switches — first allowable permutation.

Introducing project Golden Eagle. Project Golden Eagle is a new security initiative from Solid Security. Our goal with project Golden Eagle is to...

Figure 12: Spintax with global switches — second allowable permutation.

As is evident in the example, it is necessary that the project name stays consistent throughout the document. If the document mentions "Purple Phoenix" in one sentence, and then mentions "Golden Eagle" in another, the discrepancy could cause confusion.

Introducing project {Purple Phoenix|Golden Eagle}. Project {Purple Phoenix|Golden Eagle} is a new security initiative from Solid Security. Our goal with project {Purple Phoenix|Golden Eagle} is to...

Figure 13: Standard spintax equivalent.

Consider the equivalent statement with local switches in standard spintax (Fig. 13). The use of non-global switches can cause the resulting documents to have inconsistencies (Fig. 14).

Introducing project Golden Eagle. Project Purple Phoenix is a new security initiative from Solid Security. Our goal with project Purple Phoenix is to...

Figure 14: Inconsistent result from the use of non-global switches in standard spintax.

3.3. Linguistic Intelligence

Intelligent adversaries are likely to be wary of unusual verbiage found in documents that they receive. Even if the hypothetical leaker does not have access to the underlying spintax mechanism, they may choose to paraphrase content before leaking it if it sounds unnatural. If the underlying system is well-understood, any linguistic irregularities may unintentionally reveal information about the model and spintax, allowing the attacker to circumvent these measures. In order to retain the ability to trace the leak back to its source, this capability must be eliminated. It is therefore crucial that the spintax is only applied to words that are not foundational to the semantic meaning one wants to convey.

If text that is fundamentally important to the idea being communicated is arbitrarily modified, the spintaxer may fail to convey those ideas intact and violates message integrity properties, severely interfering with organizational operations. Furthermore, if text that is critical to the syntactic structure, semantic meaning, or pragmatic understanding is changed, then it may be apparent that the document has been modified, and an intending leaker may try to bypass instituted countermeasures.

As seen with the global switches example, these cues may originate from simple inconsistencies in spintax representation, but it can also come from more conceptually difficult examples, such as understanding what are acceptable variations based on the semantic and pragmatic context. For example, if one wishes to spin the colors of a military airplane, it might make sense for a plane to be green, white, grey, brown, etc. However, it is likely that a recipient would be suspicious of alterations if they read that a government plane was planned to be pink. In other contexts, pink might be a perfectly acceptable option in spintax switches detailing color specifications. Thus, it is important that there is a ruleset that outlines which words should be reasonably spintaxed, and it is important that these rules are intelligent and that they consider the possible interpretations of the resulting text. Doing this in a way that respects contextual clues is a computationally difficult natural language processing task.

3.4. Decoupling

As the field of natural language processing is rapidly advancing, it is necessary to decouple aspects of the system which might rapidly change over time (such as the linguistic rulesets and mechanisms by which they are generated) from systems which we expect to remain relatively static (such as the spinning and storage mechanisms). The use of a versatile spintax standard as an intermediate format allows for stronger rulesets to be developed over time without affecting other components of the system, ensuring that the systems proposed herein remain viable even while computational linguistics evolves.

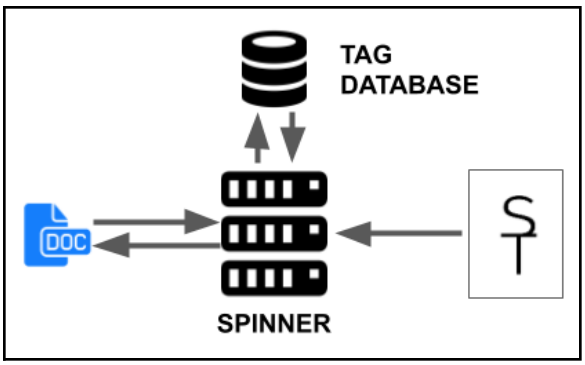

Figure 15: Two-way determinability relationship between permuted documents and tags.

3.5. Indexability Properties

Indexability is a critical property of the Solid Spintax system. Specifically, the number of permutations possible from any given spintax is always finite, countable, and thus uniquely indexable. There exists a bi-directional relationship between document permutations and "tag" identifiers (Fig. 15). A spinner can receive a spintax and be told to deterministically generate "permutation #1,501." Similarly, when presented with the generated document and original spintax, the spinner should be able to identify that the document presented was "permutation #1,501."

As a direct result of this property, rather than storing every document sent to each recipient, each document variation is indexed such that the system only needs to store an association between a tag and a recipient, along with the generating spintax, allowing for significant advances in storage efficiency. The tag can thus be thought of as more of a "roadmap" for which option a spinner should choose at every available switch rather than simply a random identifier.

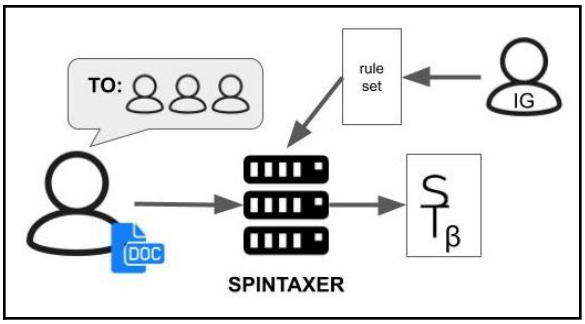

Figure 16: Role of the spintaxer aspect of the proposed Tailspin system.

4. System Overview

4.1. Spintaxer

The spintaxer component of the Tailspin system, illustrated in Figure 16, is primarily tasked with generating an initial (or Beta) spintax. The spintaxer is provided an original (source) document by the document author along with a list of intended recipients and optionally a set of attributes describing this document (for example, it's security sensitivity or clearance level). Based on the document, attributes, and recipients, a ruleset is provided or generated by an inspector which instructs the spintaxer how to transform the document into a spintax. The spintaxer then uses these components to generate a Beta spintax indicating all of the possible permutations of the document compliant with the provided ruleset.

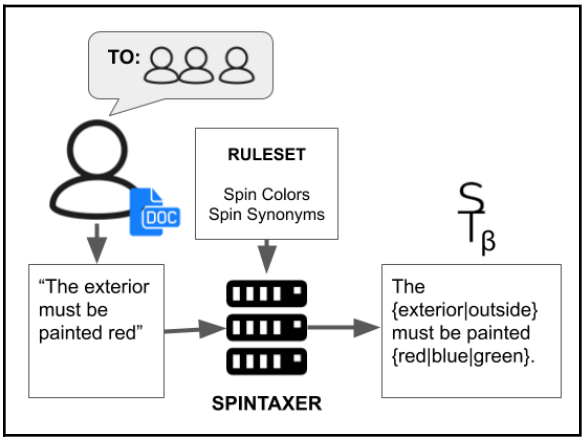

Figure 17: Spintaxer generates a beta spintax from an example ruleset and document.

An example of this process is illustrated in Figure 17. The document author provides the simple sentence "The exterior must be painted red." A ruleset is provided which includes the directions "spin colors" and "spin synonyms." In realistic applications, these rules would formally outline the exact character patterns that constitute "colors" and "synonyms." The spintaxer combines these source documents to generate the beta spintax "The {exterior|outside} must be painted {red|blue|green}."

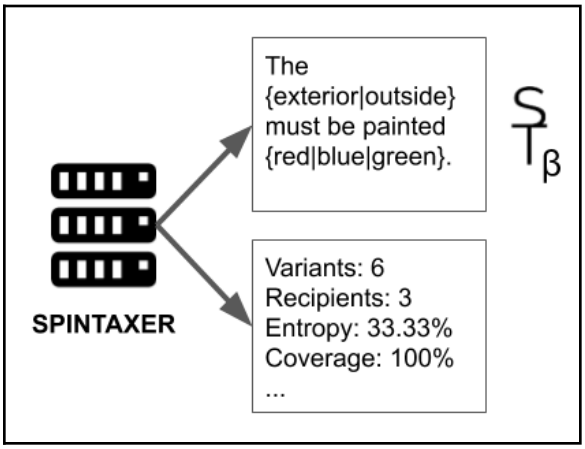

Figure 18: Spintaxer generates a series of statistics describing the generated spintax.

Figure 18 illustrates that in addition to generating a beta spintax, the spintaxer might also at this stage generate a set of statistics describing characteristics of the generated Beta spintax, particularly pertaining to its variability and retention of semantic meaning.

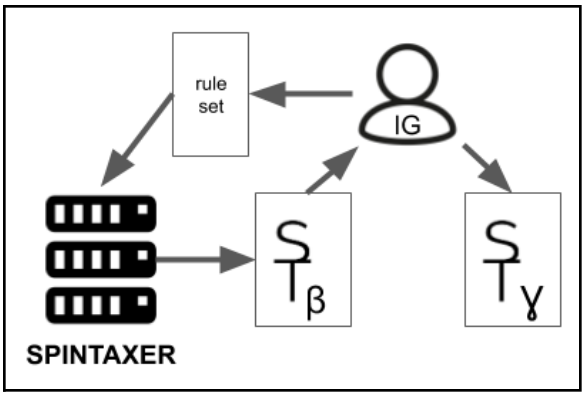

Figure 19: Inspector provides a ruleset to the spintaxer and produces a gamma spintax from the generated beta spintax.

4.2. Inspector

The inspector component of the Tailspin system is tasked with providing a ruleset to the spintaxer for use in generating a Beta spintax, and, after receiving a beta spintax and set of relevant statistics from the spintaxer, producing a final (Gamma) spintax for use by the spinner (Fig. 19).

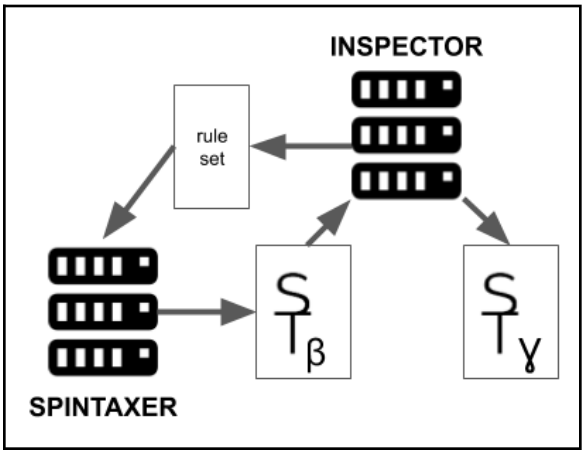

Figure 20: The inspector in this embodiment of the system is fully autonomous.

The use of a separate inspector component allows for an automated system to be used in place of a manual inspector (Fig. 20). In some implementations, a machine learning system may be used which takes the statistics generated by the spintaxer as feedback on the efficacy of generated spintaxes to generate progressively better rulesets. Such a system may be necessary to handle the communication needs of a large organization.

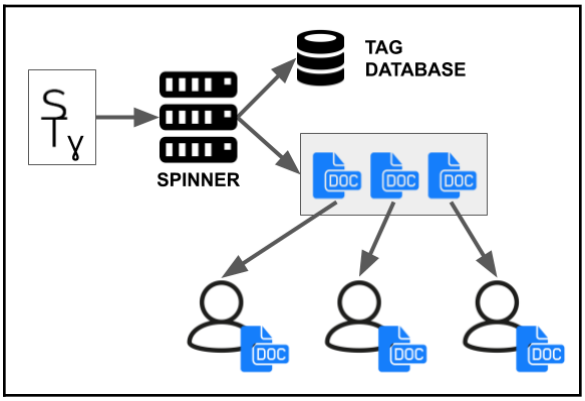

Figure 21: Spinner aspect of the Tailspin system.

4.3. Spinner

The spinner component of the Tailspin system (Fig. 21) uses the provided gamma spintax and list of recipients to generate uniquely identifiable documents and document identifiers (tags) for storage in a tag database. Per the specification of our Solid Spintax standard, there is a one-to-one correspondence between tags and document permutations, so storage of the tag is sufficient to identify the entire document text sent to each recipient. An example of this process is illustrated in Figure 22.

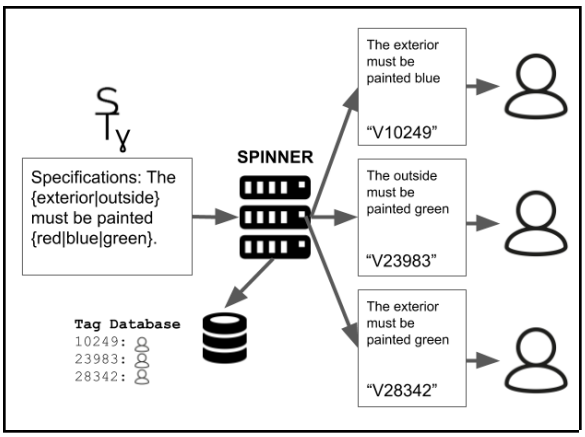

Figure 22: Example of documents being tagged and stored in a tag database.

In this example, the inspector component has provided the following gamma spintax: "The {exterior|outside} must be painted {red|blue|green}." The spinner generates multiple permutations of the document compliant with the provided spintax and sends a different version to each recipient. The tag database is then updated with records indicating which permutation was sent to which end recipient.

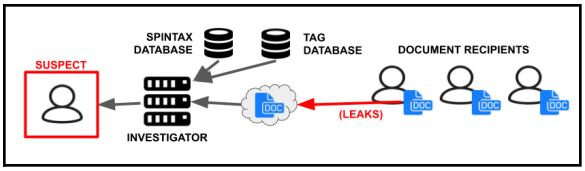

Figure 23: Investigator identifies a suspect from a leaked document.

4.4. Investigator

In the event of a leak, the investigator component is tasked with identifying a suspect (Fig. 23). The investigator first uses the spintax file stored in a spintax database along with the leaked document to recompute the document identifier tag via the processes described in the spintax section. The investigator then checks the obtained tag against the tag database to identify the recipient associated with that tag, which is ultimately the source of the leak.

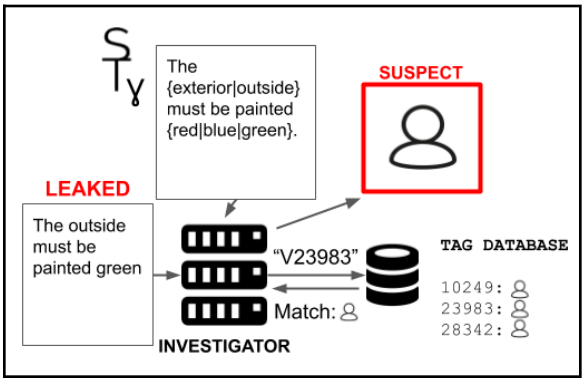

Figure 24: Investigator compares leaked document to spintax to identify a suspect.

Figure 24 illustrates an example of the investigator using a leaked document and stored syntax to identify a suspect. In the example, the leaked document contains the text "The outside must be painted green." The stored spintax specifies that "The {exterior|outside} must be painted {red|blue|green}." The investigator considers which option was taken at each switch encountered to recover the original tag, "V23…," which is then compared to the entries in the tag database to identify a particular suspect.

Approach	Content is unique	Storage Efficient	Format Agnostic
Direct	✗	✓	✓
Metadata Tagging	✗	✓	✗
Standard Spintax	✓	✗	✓
Tailspin	✓	✓	✓

Table 3: Advantages and benefits of various identification approaches.

5. Advantages and Benefits

5.1. Compliance & Auditing

Our standard requires the initial (beta) spintax to be generated according to a specified ruleset. This ruleset can be carefully constructed to ensure compliance with organizational or regulatory standards on text variability and meaningfulness. The generated spintax, as well as a series of statistics about it, are then presented to the inspector component for approval and generation of a final spintax. The "inspector" component of the system is, therefore, the de-facto compliance auditor for the system, making Tailspin a compelling option for organizations with strict compliance and auditing requirements.

5.2. Loose Coupling

As the fields of machine learning and natural language understanding are rapidly evolving, it is important that systems are able to be easily expanded upon to accommodate innovations in order to reduce the work required to bring systems up to date. It is inherent in our design of spintax that a paradigm of loose coupling is maintained. The benefit of this characteristic is that it allows for the spintax ruleset, along with the mechanism by which it is generated, to be changed without needing to change the system that applies those rules. In the event that future developments in natural language processing or machine learning facilitate improvements to the process outlined in this paper, one would not need to "reinvent the wheel" and create an entirely new system. Instead, one could simply implement these new aspects in the generation of the ruleset. Similarly, if regulatory changes impose new compliance and auditing requirements, the inspector component of the system could be adjusted accordingly without necessitating changes to other components. In the event that an unforeseen development requires an entire component, such as the inspector or investigator, to be replaced entirely, the remaining components can nevertheless be reused and should continue to interact with the new component seamlessly.

5.2. Format Agnosticism

As Tailspin operates directly on a stream of characters, it is fundamentally format agnostic. Popular text document formats like .txt, .doc, .docx, .pdf, etc. would all work reasonably with the Tailspin system as long as a mechanism to convert the document to a stream of characters and then back from a stream of characters to a document of the desired format can be devised.

5.3. Content Distinguishability

As discussed in our introduction, mechanisms for making content uniquely identifiable can be differentiated into those that primarily protect the file container in which the document is enclosed (eg. metadata tagging) and those which protect the content itself. Tailspin falls firmly in the category of protecting document content, the numerous security benefits of which are discussed in sections 1.3 and 2.

5.4. Storage Efficiency

The indexability property of the Solid Spintax standard enables significant advancements in storage efficiency. These improvements are particularly noticeable with relatively large files being sent to large groups of people, whereby a large number of document variants would be generated. Traditional methods of content randomization would require every variant of every document to be stored in a database for later identification upon a leak. Solid Spintax's indexability property allows the indexes (tags) alone to be stored, as is discussed in section 3.5.

	Standard Spintax	Tailspin
Document Size	40 KB	40 KB
Recipients	10,000	10,000
Total Switches	1000	10,000
Options per Switch	10	25
Spintax Size	102 KB	1.6 MB
Tag Size	(N/A)	6.3 KB
Spintaxes Stored	0 (0B)	1 (1.6 MB)
Tags Stored	0 (0B)	10,000 (62.5 MB)

Variations Stored	10,000 (400 MB)	0 (0B)
Total Size	400 MB	64.1 MB

Table 4: Total storage size for this paper processed by standard spintax vs. Tailspin.

Table 4 takes this paper as an example document to be sent to 10,000 recipients. The uncompressed size of the ASCII text content of this paper is approximately 40,000 bytes. Assume a switch can be inserted, on average, every 5 words. We estimate that a standard spintax document might contain 1,000 switches with 10 options each, while a Tailspin document might contain 10,000 switches with 25 options each (due to the tendency of nested switches to promote exponential growth and the large number of options available in integer switches positively skewing the average number of options per switch). We can approximate the size of the stored spintaxes as:

$$S_{spintax} = S_{orig} + 2N_{switches} + 6N_{options}$$
$$= S_{orig} + 2N_{switches}(1 + 3N_{options\ per\ switch})$$

Formula 1: Approximate size of stored spintaxes.

This formula yields a spintax size of 102KB for the standard spintax and for Solid Spintax at 1.5 MB. The tag size (in bits) is calculated in Solid Spintax as follows:

$$S_{tag} = N_{switches} ceil(log_2(N_{options\ per\ switch}))$$

Formula 2: Size of Solid Spintax tags.

This formula yields a tag size of 50,000 bits or roughly 6.3 KB. Tailspin ultimately stores the source spintax along with 10,000 tags, while a standard spintaxing approach would store no spintaxes nor tags, but 10,000 individual document variations. This ultimately results in the standard spintax method storing 400 MB of data while the Tailspin approach stores just 64.1 MB.

6. Future Work

6.1. Entropy and Signal-Noise Ratio

Future improvements could be made to ensure an appropriate ratio of linguistic signal and noise. If balanced correctly, the system could meaningfully identify even partial leaks without infringing on semantic meaning. The goal, therefore, is to maximize entropy and ensure its even distribution throughout the document while maintaining the core message to within certain thresholds. The techniques which may be applied to perform this optimization merit further study.

6.2. Partial Leaks

In the event that only a portion of the variations present in a document are leaked, it is necessary that there exists a system to nevertheless extract all meaningful information from the partial leak and potentially identify a suspect despite having incomplete data. In future publications, we will describe the modeling process that allows for partial leaks to be traced.

6.3. Incomplete Matches

If an intending leaker were to paraphrase or alter a leaked document such that they do not precisely match any known spintax variation, they may be able to circumvent the measures outlined in this paper. To prevent this, modern natural language processing techniques for calculating semantic similarity can be applied to confirmed leaks to assign probabilities that a given recipient is the source of the leak. This mechanism will be included in the novel modeling techniques outlined in future publications.

6.4. Ruleset Generation

The initial implementation outlined in this paper specifies that a manual inspector is responsible for reviewing a document, creating a spintax ruleset, and reviewing the provided beta spintax before generating a gamma spintax. However, this process does not actually necessitate a manual inspector. Future improvements could apply natural language processing techniques to generate the ruleset by doing a semantic analysis on which words are crucial to the sentence. Due to the importance of high linguistic accuracy and the extremely low tolerance for failure in critical applications, it is likely that initial embodiments of such a system would merely provide suggestions for manual review.

Nevertheless, this would still greatly reduce the workload of the inspecting body, and other use cases may not require such accuracy.

7. Conclusion

We have proposed a system to aid in the detection, prevention, and mitigation of data leaks of text documents by making the content of such documents distinct without affecting their semantic meaning. While well known steganographic techniques have been developed for inserting data into images, audio files, videos files, executable code, and other file types nearly undetectably, doing so scalably for text documents is less well understood. The steganographic techniques for these file formats share similarities in that they all involve changing some aspect of the data values in a way that is insignificant to humans, but detectable by machines. However, similar steganographic techniques have historically been difficult to accomplish with text content, as small random changes to the document text are immediately recognizable. Modern natural language processing may make this possible, and our system aims to support this operation in a practical and scalable manner.

The modern approach to addressing this problem with textual documents is to embed a unique piece of metadata within a document before delivering it to its recipient. As previously mentioned, such a method is flawed in that it only protects the file container, not the textual document itself. Embedding metadata can be trivially bypassed by moving data to a new container, a vulnerability which the Tailspin system is not susceptible to.

In future work, we will outline the methods by which partial leaks, along with leaks which do not perfectly match a document received by any particular person, can be used to build a dynamic threat model and ultimately identify a suspect.

Overall, the Tailspin method promises to improve on alternatives by offering a loosely-coupled architecture, ease of compliance and auditing, format agnosticism, and increased storage efficiency. Future work on analyzing partial leaks and incomplete matches is likely to make the standard practical for real-world application.

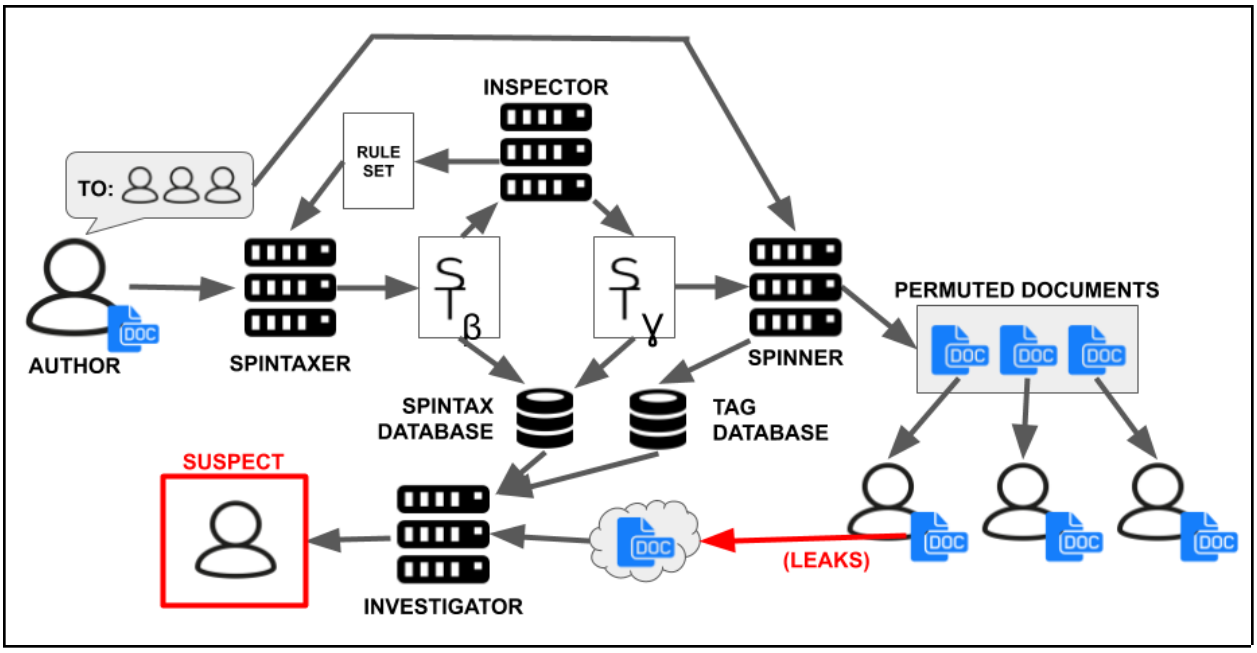

Figure 25: A complete embodiment of the Tailspin system as currently proposed.