Using Word Embeddings to Deter Intellectual Property Theft through Automated Generation of Fake Documents

ALMAS ABDIBAYEV, DONGKAI CHEN, HAIPENG CHEN, DEEPTI POLURU, and V. S. SUBRAHMANIAN, Dartmouth College

Theft of intellectual property is a growing problem—one that is exacerbated by the fact that a successful compromise of an enterprise might only become known months after the hack. A recent solution called FORGE addresses this problem by automatically generating N "fake" versions of any real document so that the attacker has to determine which of the $N + 1$ documents that they have exfiltrated from a compromised network is real. In this article, we remove two major drawbacks in FORGE: (i) FORGE requires ontologies in order to generate fake documents—however, in the real world, ontologies, especially good ontologies, are infrequently available. The WE-FORGE system proposed in this article completely eliminates the need for ontologies by using distance metrics on word embeddings instead. (ii) FORGE generates fake documents by first identifying "target" concepts in the original document and then substituting "replacement" concepts for them. However, we will show that this can lead to sub-optimal results (e.g., as target concepts are selected *without* knowing the availability and/or quality of the replacement concepts, they can sometimes lead to poor results). Our WE-FORGE system addresses this problem in two possible ways by performing a joint optimization to select concepts and replacements simultaneously. We conduct a human study involving both computer science and chemistry documents and show that WE-FORGE successfully deceives adversaries.

CCS Concepts: • **Security and privacy** • **Computing methodologies** → *Artificial intelligence*;

Additional Key Words and Phrases: AI security, fake document generation

ACM Reference format:
Almas Abdibayev, Dongkai Chen, Haipeng Chen, Deepti Poluru, and V. S. Subrahmanian. 2021. Using Word Embeddings to Deter Intellectual Property Theft through Automated Generation of Fake Documents. *ACM Trans. Manage. Inf. Syst.* 12, 2, Article 13 (January 2021), 22 pages.
https://doi.org/10.1145/3418289

1 INTRODUCTION

Intellectual property theft is a growing problem for the United States. According to a February 2020 report, the FBI is investigating more than a 1,000 cases of theft of US intellectual property carried out by just one nation state [7]. A 2019 CNBC report states, likewise, that 1 in 5 US companies feel that their Intellectual Property (IP) has been stolen by one nation state [14]. Regardless of whether

Author list is alphabetically ordered.

Parts of this work were supported by ONR grants N00014-18-1-2670 and N00014-16-1-2896.

Authors' address: A. Abdibayev, D. Chen, H. Chen, D. Poluru, and V. S. Subrahmanian (corresponding author), Dartmouth College, 09 Maynard Street, Hanover, New Hampshire, 03755; emails: {almas.abdibayev.gr, dongkai.chen.gr}@dartmouth.edu, {haipengkeeon, deeptipoluru.gr}@gmail.com, vs@dartmouth.edu.

2158-656X/2021/01-ART13 $15.00
https://doi.org/10.1145/3418289

these allegations are true or not, it is clear that IP theft is viewed as a major problem by US corporations. The problem is even more severe because results from cybersecurity firm Symantec state that there is a gap of 312 days on average from the time an enterprise is compromised by a zero-day attack and the time the compromise is discovered [1]. This provides almost a full year for an adversary who has successfully targeted an enterprise to exfiltrate valuable intellectual property.

The goal of this article is to deter IP theft. The essence of deterrence is to increase uncertainty and impose costs on the adversary—concepts pionereed by Nobel Laureate Tom Schelling [16] when fighting the Cold War. As powerfully stated by Huth [8], "a threat serves as a deterrent to the extent that it convinces its target not to carry out the intended action because of the costs and losses the target would incur."

The authors of Ref. [5] leveraged these important ideas in order to deter cyber-theft of intellectual property—specifically, technical documents such as scientific papers and patents. They proposed a framework that takes an "original" document d (i.e., the real one) as input and generates a set $\mathcal{F}$ of fake documents that are similar enough to d to be believable, but yet are sufficiently different to be incorrect. Each fake document $f \in \mathcal{F}$ is obtained by replacing certain concepts c in d with a replacement concept c'. A cyber-attacker who steals the resulting set of $|\mathcal{F}| + 1$ documents would need to spend time and effort (i.e., incur cost) to determine which of the $|\mathcal{F}| + 1$ documents is the real one. Moreover, even if the attacker decided that a document $d^\star \in \mathcal{F} \cup \{d\}$ is the real one, he would still be left in some doubt about whether he is right, i.e., whether $d^\star = d$. While the Fake Online Repository Generation Engine (FORGE) system [5] is shown to generate a believable document, it suffers from four major flaws.

- **Good ontologies are needed.** First, FORGE assumes that there is an ontology available for a given domain and that this ontology is appropriate for the domain in question. Though a number of ontologies do exist for various knowledge representation purposes, they do not exist for numerous domains, especially specialized domains. Moreover, they are financially expensive and time consuming to develop. Reference [17] discusses the costs of building ontologies. For instance, if a company like Pfizer wants to develop an ontology for a new autism drug, they would need to develop a new ontology for autism. If, on the other hand, Toshiba needs to generate an ontology related to portable hard drives, they would need to create one from scratch. This is manually intensive—taking time and effort. *In this article, we develop a general approach to identifying replacement concepts that do not require the existence of an ontology.* We thus save both time and money.
- **The best concept-replacement pairs may not be selected.** Second, FORGE first selects a set of concepts to replace in the original document d. For each selected concept, it then identifies a set of appropriate replacement concepts, which it uses to generate the fake documents. However, this is suboptimal. For instance, a concept c in the original document may be an excellent one to replace in theory, but there may be no really good replacement terms for it. Because the concepts to replace are fixed without considering the availability and/or quality of the replacement, this can lead to suboptimal choices. In contrast, we develop a *single unified method* that evaluates concepts and their possible replacements simultaneously, choosing the (concept, replacement) pairs that work best.
- **The replacements chosen for a given concept may be deterministic.** Replacing a concept c by the same replacement concept c' across multiple documents may make it easier for an adversary to detect the replacement concepts. One of the two algorithms developed in this article makes replacements via a stochastic choice, which is non-deterministic—and, in fact, our results show that this algorithm is better at deceiving the adversary.
- **The set $\mathcal{F}$ of fake documents generated may be very similar.** FORGE does not ensure that there is *diversity* in the set of fake documents generated. For instance, if we wish to

generate 99 fake documents, it is possible that FORGE generates 99 fakes that all differ very minimally from each other because the set of (concept, replacement) pairs used to generate each fake document vary minimally.

We, therefore, propose the new Word-Embedding based Fake Online Repository Generation Engine (WE-FORGE) architecture to address these three major shortcomings of FORGE via the following important innovations:

- —We develop an architecture that merges word embeddings [3, 12] and clustering [10] in order to identify potential replacements for concepts. In particular, given a concept c in the original document, we develop embeddings for the concepts (in embeddings, each concept is represented as a vector) and then cluster the resulting set of word embeddings (and, hence, the concepts themselves). The idea is that if a concept c' is in the same cluster as c, then c' could serve as a possible replacement for c, with the probability of c' being a good replacement for c being inversely proportional to the distance between c and c' according to a distance metric chosen by the system security manager.
- —We pose the problem of selecting the best concept to replace and the best replacement as two Joint Concept Replacement (JCR) *problems* (JCR-Implicit and JCR-Explicit), in which the concept selected for replacement depends upon the quality of the replacements that are available. We additionally show that both the JCR problems are NP-hard.
- —In order to ensure diversity in the set of fake documents generated, we incorporate a regularization term in the objective function of the JCR problems to ensure diversity. The level of diversity can be easily regulated on an application by application basis.
- —We evaluate WE-FORGE via detailed experiments involving human subjects (with Institutional Review Board (IRB) authorization). Our experiments on a chemistry patent dataset and a computer science patent dataset show that WE-FORGE performs well on real-world technical documents. In particular, we show that the implicit version of WE-FORGE outperforms the explicit version of WE-FORGE. Moreover, both WE-FORGE versions outperform FORGE. These results show that WE-FORGE is clearly superior to FORGE [5], even if we do not count the time and cost savings because WE-FORGE does not need ontologies.

Figure 1 shows a paragraph from US Patent US20020042540A1 [11]. The original paragraph is shown with highlighted words—these are concepts selected for replacement. We show one fake document (paragraph) generated by the explicit version of JCR and the implicit version—the replacement terms are highlighted in red.[1]

The remainder of this article is organized as follows. Section 2 discusses related work. Section 3 introduces the general WE-FORGE architecture, followed by the introduction of the detailed optimization-based fake document generation approach in Section 4. We present computational complexity results of the formulated optimization problems in Section 5, and empirical results in Section 6. Section 8 concludes the article.

2 RELATED WORK

A number of past efforts used "honey" files (e.g. files called PASSWORDS.TXT) to attract attackers — once an attacker accesses such a file, the access is logged and securit officers are notified. [24].

[1]In fact, some substitutions are not perfect. In our WE-FORGE system based on this article, the author of a document d is expected to use WE-FORGE to generate fakes of a given document. The system shows him/her the concepts c chosen for replacement and the suggested replacement c' for a given fake version f of the document. It also shows him a set $Cand(c)$ of other candidate replacements for c. The user can choose to use c' or any member of $Cand(c)$ as the replacement for c. He/she may also type in a completely different replacement.

Patent: US20020042540A1
Original:
In general formula (I) for diphosphines, R 1 to R5 can also represent a saturated, unsaturated or aromatic heterocyclic radical, in particular comprising 5 or 6 atoms in the ring including 1 or 2 heteroatoms such as the nitrogen, sulphur and oxygen atoms; the carbon atoms of the heterocycle being optionally substituted. R1 to R5 can also represent a polycyclic heterocyclic radical defined as being either a radical constituted by at least 2 aromatic heterocycles or heterocycles not containing at least one heteroatom in each ring and forming together ortho- or ortho- and peri-condensed systems or a radical constituted by at least one aromatic or non-aromatic hydrocarbon ring and at least one aromatic or non-aromatic heterocycle together forming ortho- or ortho- and peri-condensed systems; the carbon atoms of said rings being optionally substituted.

Explicit:
In general formula (I) for diphosphines, R 1 to R5 can also represent a saturated, unsaturated or aromatic heterocyclic radical, in particular comprising 5 or 6 atoms in the ring including 1 or 2 heteroatoms such as the nitrogen, gypsum and oxygen atoms; the carbon atoms of the diamine being optionally substituted. R 1 to R5 can also represent a polycyclic heterocyclic radical defined as being either a radical constituted by at least 2 aromatic heterocycles or heterocycles not containing at least one trifluoromethyl in each ring and forming together ortho- or ortho- and peri-condensed systems or a radical constituted by at least one aromatic or non-aromatic hydrocarbon ring and at least one aromatic or non-aromatic diamine together forming ortho- or ortho- and peri-condensed systems; the carbon atoms of said rings being optionally substituted.

Implicit:
In general formula (I) for diphosphines, R 1 to R5 can also represent a saturated, unsaturated or aromatic heterocyclic radical, in particular comprising 5 or 6 atoms in the ring including 1 or 2 heteroatoms such as the nitrogen, cobalt and oxygen atoms; the carbon atoms of the amide being optionally substituted. R 1 to R5 can also represent a polycyclic heterocyclic radical defined as being either a radical constituted by at least 2 aromatic heterocycles or heterocycles not containing at least one ligand in each ring and forming together ortho- or ortho- and peri-condensed systems or a radical constituted by at least one aromatic or non-aromatic hydrocarbon ring and at least one aromatic or non-aromatic amide together forming ortho- or ortho- and peri-condensed systems; the carbon atoms of said rings being optionally substituted.

Fig. 1. Sample of Fake Paragraph Generation using WE-FORGE[E] and WE-FORGE[I] on a paragraph from from US Patent US20020042540A1 [11]. Highlighted portions show some (not all) important modified parts.

A related effort [4] develops the D^3 system which uses decoy documents with attractive names to lure insiders who pose a threat. Another insider threat detection system generates decoy software by modifying genuine software into bogus programs using obfuscation techniques [13]. The bogus programs contain a beacon, which indicates when and where the decoy is accessed. Reference [19] translates genuine documents into a foreign language and sprinkles untranslatable nouns within it. These efforts are very different from the work reported here in several respects: (i) they do not develop NLP-based methods to automatically generate fake files at scale; and (ii) they do not make the tradeoff between believability so that the adversary believes that a fake file is real and incorrectness—so that there is some difference between the original file and the fake one—that we do.

Closer to our work is Ref. [22], which proposes the concepts of "Canary Files" and "Canary File management system" for network intrusion detection. A canary file is a fake document placed among real documents in order to rapidly detect unauthorized data access, copying, or modification. Reference [23] states that fake documents must: (1) be enticing, (2) be realistic, (3) minimize disruption, (4) be adaptive, (5) provide scalable protective coverage, (6) minimize sensitive artifacts and copyright infringement, and (7) contain no characteristics that distinguish them from

the real document. In particular, it points out that the key is to generate canary files that are believable. Reference [21] further states that it usually takes several months to produce a realistic decoy by examining all the topics that can be manipulated to generate fake documents. Reference [20] proposes using a separate module to generate decoy documents for each topic, which is time-consuming and difficult to scale. Whereas these works have identified the tradeoff between scalability and believability of fake document generation, they pay less attention to the tradeoff of sensitive artefacts minimization and believability.

These past efforts have several limitations. (i) First, they do not specify how to *automatically* choose the topics/concepts in the original document that should be replaced in order to generate fake documents. (ii) They do not address the goal of generating fake *technical* documents. Rather, they focus on changes to numeric data such as credit card numbers, social security numbers, ATM PIN codes, as well as addresses and other structured fields. In a nutshell, they do not do any natural language processing, which is essential to generate fake technical documents, which is the key goal of this article. (iii) They do not capture the fact that we must regulate the fake documents to be "close enough" to the original to make the fakes believable, but sufficiently "far enough" to make them likely to be wrong. (iv) Finally, none of the above efforts perform a human evaluation to assess whether humans find the resulting fakes believable or not.

This article builds upon the recent FORGE system [5] that addresses all of the four limitations (i)–(iv) mentioned above. FORGE builds a multi-layer graph of the concepts in a real document and computes certain "meta-centrality" measures on the multi-layer graph to select concepts in the original document to replace. Once a concept is selected for replacement, FORGE requires an ontology pertinent to the domain of the documents in order to select a suitable replacement term—for example, if the fake documents are being developed by a company creating pacemakers for the heart, then the ontology may relate to biomedical heart devices. We make three major improvements over FORGE. First, appropriate ontologies may not always be available—thus, even if ontologies exist for medicine in general, ones specifically focused on biomedical heart devices may not. Developing such ontologies from scratch involves multi-disciplinary teams (e.g., cardiologists, biomedical engineers, computer scientists) and can take a lot of time and effort [17]. Our WE-FORGE system avoids this altogether. Second, FORGE selects concepts to replace in the original document without considering what replacements might exist for them. This is a bit like a chef in a restaurant deciding to replace some dishes on the menu (similar to concepts in the original document) without knowing what ingredients he/she has in the kitchen. WE-FORGE decides which concepts to replace in the original document by considering *both* what replacements are possible, and whether applying the replacements will satisfy the competing goals of generating fake documents that are sufficiently believable and different from the original one. Third, WE-FORGE ensures that different fake documents all look somewhat different rather than looking very similar to each other. Finally, we show that WE-FORGE beats out FORGE from a performance perspective.

3 WE-FORGE ARCHITECTURE

Consider a major corporation (e.g., Honeywell Corporation) that produces a huge amount of intellectual property every year in a very wide set of disciplines. According to Wikipedia, Honeywell develops products in "Aerospace, Building Technologies, Performance Materials & Technologies (PMT), and Safety & Productivity Solutions (SPS)" [2]. The company produces a huge range of products including cockpit instruments, aircraft guidance systems, home and industrial thermostats, software, air conditions, air purifiers, and much more. Scientists, product designers, and engineers working for Honeywell might generate all kinds of technical documents covering a wide range

[2] https://en.wikipedia.org/wiki/Honeywell.

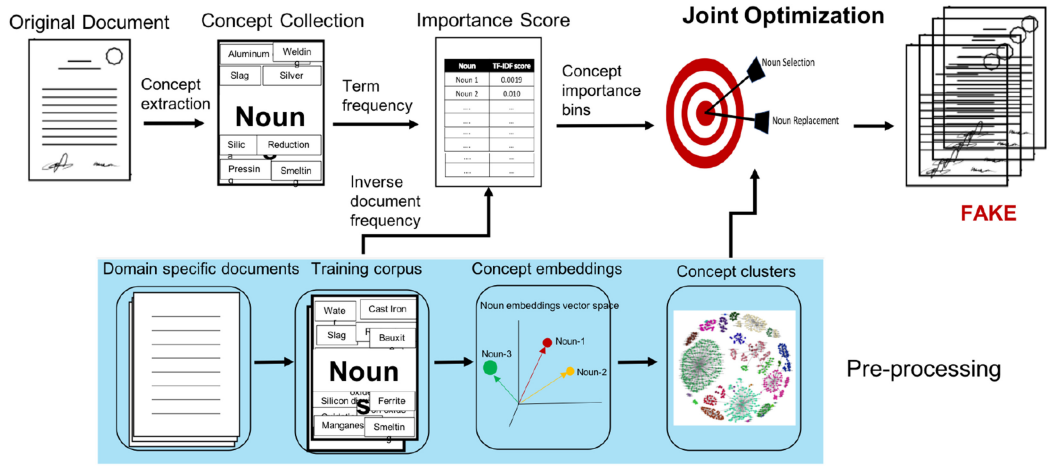

Fig. 2. A flowchart of our proposed approach.

of territory. Any fake document generation system must support the diversity of products about which fake documents need to be generated. Figure 2 shows the architecture of the proposed WE-FORGE architecture. We will now describe how a company such as Honeywell might use our architecture.

3.1 Pre-Processing

Before any fake documents are generated, some pre-processing steps need to be taken.

(1) **Building Domain Specific Corpora:** The first step is to build a corpus of domain-specific documents. In the example company, Honeywell, they would need to generate one corpus of documents about cockpit systems, another corpus about aircraft guidance systems, another one on home thermostats, and so forth. Building such corpora is quite easy—for instance, by doing searches on Google Patents or on Google Scholar or from within their own enterprise file systems. It took a script approximately ~ 10 hours to run and download 10K Computer Science documents from Google Patents. In practice, the original corpora are already in place of the companies/organizations so that such a step is not actually required. *It is important to note that our approach to fake document generation does not require any manual annotation (cf. Figure 2.)*

(2) **Learning Word Embeddings:** Next, we automatically learn word embeddings for each word in the documents. A word embedding associates a numeric vector with each word. Intuitively, given a word, we can associate a co-occurrence vector with w that captures some aspect of co-occurrence of the word w with other words w' in the vocabulary. These co-occurrence vectors can then be mapped to a lower dimensional space generating the final vector (i.e., embedding) associated with word w. We use an extension of the well-known *Word2Vec* framework [12] that learns word embeddings of sub-words [3]. Thus, at the end of this step, each n-gram from the domain-specific corpus of interest is reduced to a numeric "embedding" vector. We use v_c to denote the word vector associated with an n-gram c. Thus, word embedding associates a unique vector v_c with each concept $c \in C$. Note that learning word embeddings for 10K CS patents (with batch size 1024; 100 epochs on a 1080 Ti gpu) takes approximately 40 hours of runtime, which is very reasonable.

Table 1. Silhouette Scores of Different k Values

k	1,000	1,500	2,000	2,500	3,000	3,500	4,000	4,500	5,000
Score	0.0124	0.0113	0.0109	0.0046	0.0051	0.0072	0.0063	0.0063	0.0066

(3) **Word Pruning:** Once a domain specific document corpus has been constructed above, we can use **part of speech (POS)** tagging tools such as Natural Language Toolkit (NLTK) [2] to automatically eliminate certain types of words from consideration of being replaced. Such eliminated words can include: stop words (e.g., "the", "and"), words belonging to certain parts of speech such as adverbs (e.g., "very", "really"), adjectives (e.g., "good", "intelligent"), and even verbs (e.g., "buy", "said") because such words are very unlikely to contribute to the technical content of a document. Our WE-FORGE system prunes out all words other than nouns, but we note that this is not the only choice—instead, one could use bigrams or trigrams rather than just a single word. In this article, we focus on generating fake *technical* documents. As adverbs and adjectives are mostly used to express sentiment (e.g., "X was very bad", "X really hated Y") [18], changing them in technical documents is unlikely to change the technical content of the document. However, changing such adverbs and adjectives and verbs in, say, opinion reports (e.g., those produced by think tanks or business consultancies) could be important. We leave this for future work.

(4) **IDF Computation:** For each concept c, we can compute the inverse document frequency of that concept or word from the corpus [6].

(5) **Clustering Word Embeddings:** The final pre-processing step clusters words together. The idea is that if a word/concept is in a cluster, then any member of that cluster could potentially be a replacement of that word. Because each word is now represented as an embedding vector, we can generate clusters of these vectors in order to cluster the words together. In this article, we use standard k-means clustering [10] with Euclidean distance in order to generate clusters. To select the best k value, we use the well-known concept of silhouette score [15] to evaluate the clustering quality. Table 1 shows the results of our analysis. We choose k = 1,000 for the word embedding clusters because it has the largest silhouette score. After clustering, we can obtain a feasible candidate replacement set $\mathcal{FC}(c)$ for each concept c.

It is important to note that all of these pre-processing steps may be be performed before generating any fake documents. A company (e.g., Honeywell) might execute these steps for different product lines—for instance, they may execute these four steps for generating fake documents about cockpit instruments, and they may separately execute these four steps for generating fake documents about thermostats. Another company (e.g., a drug company like Pfizer) might generate these steps by following exactly the same steps—but, of course, the documents and embeddings and clusters they derive will be completely different.

Because the pre-processing techniques used are all standard off the shelf techniques, this article will primarily focus on the techniques used to find the best way to generate fake documents after pre-processing is complete.

3.2 Operational Use

The top part of Figure 2 shows the architecture of our system during the operational phase (i.e., after pre-processing). Once a user has created an "original" technical document d, our framework performs the following steps.

(1) **Extracting Key Concepts:** First, we extract all key concepts from d. In the current implementation of WE-FORGE, key concepts are nouns. However, these can be replaced

with more complex notions of key concepts (e.g., "hot dog" could be a key concept whose semantics is different from both the words "hot" and "dog").

(2) **TFIDF Computation:** Next, we compute the well-known term-frequency, inverse document frequency (TFIDF) metric [6, 9] for all key concepts. Note that this measure can be replaced by other metrics of the importance of concepts. Moreover, we note that the IDF of all documents was already computed during the pre-processing step, so this step only involves computing term frequencies and multiplying them with the pre-computed IDFs. We then cluster concepts according to the TFIDF measure in bins.[3] We use $\xi(c, d, \mathcal{D})$ to denote the TFIDF of concept c in document d w.r.t. corpus $\mathcal{D}$.

(3) **Binning:** We then place the concepts into *Concept importance Bins*. We sort the concepts in document d in ascending order of TFIDFs and then evenly group the concepts into n bins $B_1, \ldots, B_n$ based on their concept importance values such that for all $i, j \in 1, \ldots, n$ where $i < j$, and $c \in B_i, c' \in B_j$, we have $\xi(c, d, \mathcal{D}) \leq \xi(c', d, \mathcal{D})$. *When $d, \mathcal{D}$ are clear from context, we will abuse notation and simply write ξ_c to denote $\xi(c, d, \mathcal{D})$.*

(4) The most novel part of the article is the part that finds the concepts c to replace in document d and their corresponding replacements c'. This is done by solving a joint optimization problem described in Section 4.

(5) Finally, once a set $\{(c_1, c_1'), \ldots, (c_h, c_h')\}$ of concepts c_i and their replacements c_i' are discovered above, the replacements are made.

4 AUTOMATICALLY GENERATING THE FAKE DOCUMENTS

In this section, we show how to use the concepts defined in the preceding sections in order to automatically generate a set $\mathcal{F}$ of fake documents. We propose two methods, each of which involves solving an optimization problem. In both methods, we start by assuming that we have a directory with $\mathcal{F}$ copies of the original document d. Each $f \in \mathcal{F}$ will be used to create a fake version of d.

4.1 Explicit Joint Concept Replacement—WE-FORGE[E]

In this section, we develop a natural nonlinear integer optimization problem that determines how to create a set $\mathcal{F}$ of fake documents. We then develop an equivalent linear version.

Given an original document d, we first create $|\mathcal{F}|$ copies of d. The integer program below then provides a method to identify pairs (c, c') of concepts such that c is in the original document d and $c' \in \mathcal{F}C(c)$ is a feasible replacement concept for c in file $f \in \mathcal{F}$. We achieve this by using an integer variable $X_{c,c'f} \in \{0, 1\}$, which is set to 1 if the replacement for c in file $f \in \mathcal{F}$ is c' and 0 otherwise. The integer optimization problem described in Equation (1) assumes that the only concepts that are replaced are from a set $\mathcal{B}$ of bins selected by the system security officer. Our later experiments will involve experiments with different bins $\mathcal{B}$.

We first explain the constraints in integer program Equation (1), and then explain the objective function.

Explanation of Constraints. Constraint (i) indicates that a concept in the selected set $\mathcal{B}$ of bins should be replaced at least μ times to ensure that fake documents are sufficiently different from d. The selection of a set of bins suggests that the replacement concepts c' for c should not be "too close" to the original c nor should they be "too far" away. Intuitively, this constraint specifies a range of concept importance values. For example, if $n = 10$, and $\mathcal{B} = \{B_8, B_9\}$, the concept

[3]Formally, *term frequency* $tf(c, d)$ is defined as the number of times a concept c occurs in a certain document d, i.e., $tf(c, d) = f_{c,d}$. *Inverse document frequency* $idf(c, \mathcal{D})$ captures the rarity of a concept across all documents in the corpus $\mathcal{D}$. It is defined to be: $idf(c, \mathcal{D}) = \log \frac{|\mathcal{D}|}{|d \in \mathcal{D}: c \in d|}$. TFIDF is then defined to be: $\xi(c, d, \mathcal{D}) = tfidf(c, d, \mathcal{D}) = tf(c, d) \cdot idf(c, \mathcal{D})$.

$$
\begin{aligned}
\min \quad & \left(\sum_{f \in \mathcal{F}} \sum_{c \in d} \sum_{c' \in \mathcal{FC}(c)} dist(c, c') \cdot \xi_c \cdot X_{c,c',f} \right) - \\
& \left(\lambda \sum_{c \in d} \sum_{f \in \mathcal{F}} \sum_{f' \in \mathcal{F}} \Big| \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} \Big| \right) \\
\text{subject to} \quad & (i) \quad \sum_{f \in \mathcal{F}} \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} \geq \mu, \forall c \in \mathcal{B} \\
& (ii) \quad \sum_{c \in d} \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} \geq N_l, \ \forall f \in \mathcal{F} \\
& (iii) \quad \sum_{c \in d} \sum_{c' \in \mathcal{FC}(c)} dist(c, c') X_{c,c',f} \geq T_f, \ \forall f \in \mathcal{F} \\
& (iv) \quad X_{c,c',f} = 0, \ \forall c \notin \mathcal{B}, \forall c' \in \mathcal{FC}(c), f \in \mathcal{F} \\
& (v) \quad X_{c,c',f} \in \{0, 1\}, \ \forall c \in d, c' \in \mathcal{FC}(c), f \in \mathcal{F} \\
& (vi) \quad \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} \leq 1, \forall c \in d, f \in \mathcal{F}
\end{aligned}
\tag{1}
$$

Fig. 3. WE-FORGE[E] Integer Optimization Problem. Parameters: $\mu, \lambda > 0$ are constants; N_l is a constant that describes a lower bound on the number of concepts to be replaced when generating a fake document; T_f is a constant that describes a lower bound on the sum of the distances of replacement concepts used in a fake document from the original concepts; $\mathcal{B}$ is a set of bins.

importance values (i.e., TFIDF scores) must be within the 70% $\sim$ 90%th percentile of all concept importance values. Constraint (iv) says that if a concept c is not in the selected bins $\mathcal{B}$, then it cannot be replaced in any of the files. Constraint (ii) requires that at least N_l concepts must be replaced in any of the fake documents $f \in \mathcal{F}$. Constraint (iii) says that for each fake document $f \in \mathcal{F}$, the sum of distances of selected concepts and their replacements should not be smaller than a threshold T_f (so that the fake document will be significantly different from the original document in order to make important information in the original document "wrong"). Constraint (v) says the variables are binary. Constraint (vi) ensures that for any given fake document f and any given concept c, at most one of the concepts in the feasible candidate set $\mathcal{FC}(c)$ is used to replace c. Note that $\lambda, \mu, N_l, T_f, \mathcal{B}$ are parameters that are specified by the system manager. For instance, in our Honeywell example, the system security officer for Honeywell would specify these explicitly.

The reader may wonder about a couple of things. For instance, constraint (ii) says that at least N_l concepts in the original document must be replaced but does not, for instance, place an upper bound on the number of concepts replaced. Similarly, Constraint (iii) requires that in any given fake file f, the sum of the distances between a concept c and its replacement c' must exceed T_f, but provides no upper bound. The reason such constraints are not specified explicitly is because they are effectively enforced in our objective function, which we now discuss below.

Explanation of Objective Function. The objective function contains two terms. The first term (the triple summation) in the objective function says that we want to replace a concept that has feasible candidate replacement concepts that are "close" to it. At the same time, we want to minimize the sum of the TFIDFs of the selected concepts, subject of course to the requirement in constraint (iv) that they are from the set $\mathcal{B}$ of bins (which ensures that they are important enough). This enables us to make the generated fake documents as indistinguishable from the original document as possible

as long as they belong to the bins $\mathcal{B}$ (which effectively regulates how "far away" a replacement concept can be). Note that the second term of the objective function looks at every triple of the form (c, f, f'), where c is a concept in the original document and f, f' are fake files. The summation in this term tries to maximize the difference between whether the concept c is replaced in both files f, f'. We want this summation $\sum_{c \in d} \sum_{f \in \mathcal{F}} \sum_{f' \in \mathcal{F}} |\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}|$ to be as large as possible in order to ensure diversity of concepts that are replaced across all the fake documents generated. We multiply this summation by $(-\lambda)$ in order to achieve this.

Linearize objective function. Unfortunately, the objective function in the integer optimization problem solved by WE-FORGE[E] is nonlinear because of the presence of the absolute value in the second regularization term of the objective function. As a consequence, this problem may be hard solve. To avoid this non-linear objective function, we introduce a set of binary auxiliary variables $Y_{c,f,f}$, where for each $c \in d, f, f' \in \mathcal{F}$:

$$\begin{aligned} Y_{c,f,f'} &\leq 2 - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} \\ Y_{c,f,f'} &\leq \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} \\ Y_{c,f,f'} &\in \{0,1\} \end{aligned} \tag{2}$$

The new objective function in Equation (1) is now:

$$\min \sum_{f \in \mathcal{F}} \sum_{c \in d} \sum_{c' \in \mathcal{FC}(c)} dist(c,c') \cdot \xi_c \cdot X_{c,c',f} - \lambda \sum_{c \in d} \sum_{f \in \mathcal{F}} \sum_{f' \in \mathcal{F}} Y_{c,f,f'} \tag{3}$$

The following proposition guarantees the equivalence of the original and the linearized objective functions:

Proposition 1. *The objective function in Equation (3) combined with Equation (2) is equivalent to the objective in Equation (1).*

Proof. It suffices to prove that for each $c \in d, f, f' \in \mathcal{F}$, the objective $|\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}|$ is equivalent to $Y_{c,f,f'}$ with the constraints in Equation (2). We prove it by considering the following two cases.

First, when $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} = \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}$. Due to constraint (vi), this means $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} = \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} = 0$ or $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} = \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} = 1$. In this case, we have $|\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}| = 0$. At the same time, Equation (2) becomes:

$$\begin{aligned} Y_{c,f,f'} &\leq 0 \\ Y_{c,f,f'} &\in \{0,1\}, \end{aligned}$$

which indicates that $Y_{c,f,f'} = 0$. Therefore, $|\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}| = Y_{c,f,f'}$.

Second, when $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} \neq \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}$, i.e., $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} = 1$, $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} = 0$ or $\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} = 0, \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'} = 1$. In this case, we have $|\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}| = 1$. At the same time, Equation (2) becomes:

$$\begin{aligned} Y_{c,f,f'} &\leq 1 \\ Y_{c,f,f'} &\in \{0,1\} \end{aligned}$$

Because we are minimizing the negative of $Y_{c,f,f'}$, we have $Y_{c,f,f'} = 1$. Therefore, $|\sum_{c' \in \mathcal{FC}(c)} X_{c,c',f} - \sum_{c' \in \mathcal{FC}(c)} X_{c,c',f'}| = Y_{c,f,f'}$ still holds in this case. □

With the linearization, the explicit integration optimization WE-FORGE[E] in Equation (1) is re-written as follows.

$$
\begin{aligned}
&\min \sum_{f\in\mathcal{F}}\sum_{c\in d}\sum_{c'\in\mathcal{FC}(c)} dist(c,c')\cdot\xi_c\cdot X_{c,c',f} - \lambda\sum_{c\in d}\sum_{f\in\mathcal{F}}\sum_{f'\in\mathcal{F}} Y_{c,f,f'}\\
\text{subject to}\quad &(i)\quad \sum_{f\in\mathcal{F}}\sum_{c'\in\mathcal{FC}(c)} X_{c,c',f}\geq\mu,\forall c\in\mathcal{B}\\
&(ii)\quad \sum_{c\in d}\sum_{c'\in\mathcal{FC}(c)} X_{c,c',f}\geq N_l,\ \forall f\in\mathcal{F}\\
&(iii)\quad \sum_{c\in d}\sum_{c'\in\mathcal{FC}(c)} dist(c,c')X_{c,c',f}\geq T_f,\ \forall f\in\mathcal{F}\\
&(iv)\quad X_{c,c',f}=0,\ \forall c\notin\mathcal{B},\forall c'\in\mathcal{FC}(c),f\in\mathcal{F}\\
&(v)\quad X_{c,c',f}\in\{0,1\},\ \forall c\in d,c'\in\mathcal{FC}(c),f\in\mathcal{F}\\
&(vi)\quad \sum_{c'\in\mathcal{FC}(c)} X_{c,c',f}\leq 1,\forall c\in d,f\in\mathcal{F}\\
&(vii)\quad Y_{c,f,f'}\leq 2-\sum_{c'\in\mathcal{FC}(c)} X_{c,c',f}-\sum_{c'\in\mathcal{FC}(c)} X_{c,c',f'},\ \forall c\in d,f,f'\in\mathcal{F}\\
&(viii)\quad Y_{c,f,f'}\leq \sum_{c'\in\mathcal{FC}(c)} X_{c,c',f}-\sum_{c'\in\mathcal{FC}(c)} X_{c,c',f'},\ \forall c\in d,f,f'\in\mathcal{F}\\
&(ix)\quad Y_{c,f,f'}\in\{0,1\},\ \forall c\in d,f,f'\in\mathcal{F}
\end{aligned}
\tag{4}
$$

Fig. 4. Integer Linear Program Solved by WE-FORGE[E].

4.2 Implicit Joint Concept Replacement—WE-FORGE[I]

The integer linear program solved by WE-FORGE[E] can be huge. A single patent document can have well over a thousand concepts, each of which may have say 20 potential replacements on average. If we want 100 fake versions of each document, i.e., $|\mathcal{F}| = 100$, then we would have $1{,}000 \times 20 \times 100 = 2M\, X_{c,c',f}$ variables alone. The total number of constraints is also huge, leading to an enormous integer linear program, which we prove to be NP-hard in Section 5. In this section, we develop a variation of the Integer Linear Programming (ILP) solved by WE-FORGE[E]. We call this variant WE-FORGE[I] (or "Implicit" WE-FORGE), which is much smaller and, hence, easier to solve in practice. WE-FORGE[I] implicitly determines which candidate $c' \in \mathcal{FC}(c)$ would be used to replace the selected concept c in the optimization procedure. We achieve this by incorporating the effect of concept replacement using the average distance $\overline{dist}(c, \mathcal{FC}(c))$ as shown in the implicit integration optimization below.

Note that here, we have an integer variable $X_{c,f} \in \{0, 1\}$ for every concept-fake file pair (c, f), where $X_{c,f}$=1 intuitively means that concept $c \in d$ will be replaced when generating the fake document $f \in \mathcal{F}$.

Explanation of the Objective Function. As in the case of WE-FORGE[E], the objective function has two parts. The first part uses $\overline{dist}(c, \mathcal{FC}(c)) = \frac{\sum_{c'\in\mathcal{FC}(c)} dist(c,c')}{|\mathcal{FC}(c)|}$ to denote the average distance between concept c and the feasible candidates in the set $\mathcal{FC}(c)$. The objective function's first part sums up the products of the TFIDF ξ_c of c and the average distances between concepts c selected for replacement and the average replacement distance to $\mathcal{FC}(c)$. Unlike the optimization problem used by WE-FORGE[E] (cf. Equation (1)), we implicitly encode concept replacement in the joint optimization problem. The second part is the same regularization term used by WE-FORGE[E], which ensures that different fake files have some diversity in the concepts replaced.

$$
\begin{aligned}
\min \quad & \sum_{f\in\mathcal{F}}\sum_{c\in d}\overline{dist}(c,\mathcal{FC}(c))\cdot\xi_c\cdot X_{c,f} - \lambda\sum_{c\in d}\sum_{f\in\mathcal{F}}\sum_{f'\in\mathcal{F}}|X_{c,f}-X_{c,f'}| \\
\text{subject to} \quad & (i) \quad \sum_{f\in\mathcal{F}} X_{c,f} \geq \mu, \forall c\in\mathcal{B} \\
& (ii) \quad \sum_{c\in d} X_{c,f} \geq N_l,\ \forall f\in\mathcal{F} \\
& (iii) \quad \sum_{c\in d}\overline{dist}(c,\mathcal{FC}(c))X_{c,f} \geq T_f,\ \forall f\in\mathcal{F} \\
& (iv) \quad X_{c,f}=0, \forall c\notin\mathcal{B}, \forall f\in\mathcal{F} \\
& (v) \quad X_{c,f}\in\{0,1\}, \forall c\in d, f\in\mathcal{F}
\end{aligned}
\tag{5}
$$

Fig. 5. Non-Linear Integer Program of WE-FORGE[I].

Explanation of the Constraints. Constraints (i) to (v) have underlying intuitions that correspond precisely to those in Equation (1)—except that they are modified to use $X_{c,f}$ variables instead of $x_{c,c',f}$ variables, which leads to a huge decrease in the number of variables. Suppose the total number of concepts in the concept importance bin $\mathcal{B}$ is M, the number of concepts in the feasible candidate replacement set $\mathcal{FC}(c)$ for each concept $c = 1,\ldots,M$ is n_c, and the total number of fake documents $\mathcal{F}$ is F; then, the total number of variables in the linearized versions of WE-FORGE[E] and WE-FORGE[I] are respectively

$$\left(\sum_{c=1}^{M} n_c\right) * F + M * F^2 \sim O(M * N * F + M * F^2)$$

and

$$M * F + M * F^2 \sim O(M * F^2),$$

where $N = \max\{n_c | c\in\mathcal{B}\}$ is the maximum number of feasible candidate replacement concepts for a single concept. We can see that due to the term $M * N * F$, the number of variables in WE-FORGE[E] would be much larger than that of WE-FORGE[I], especially when N is large. Since the total number of constraints are linear w.r.t. the number of variables, the same result applies to the number of constraints.

$\lambda, \mu, N_l, T_f, \mathcal{B}$ are parameters that should be specified by the system security officer. WE-FORGE[I] solves the optimization problem shown in Equation (5) to identify which concepts c to replace. It then selects the replacement for c from the following distribution over $\mathcal{FC}(c)$:

$$p(c') = \frac{e^{dist(c,c')}}{\sum_{c''\in\mathcal{FC}(c)} e^{dist(c,c'')}}.$$

Intuitively, this distribution makes concepts near c less likely to be chosen than those further away from c. One may think this is counter-intuitive—after all, we want to pick concepts c' that are not too far away from c. However, note that we are choosing concepts c in the optimization problem solved by WE-FORGE[E] by minimizing the average distance between c and the feasible candidates in the objective function; so, on average, the feasible candidates in $\mathcal{FC}(c)$ should not be too far away from c.

Linearize objective function. As in the case of WE-FORGE[E], the optimization problem solved by WE-FORGE[I] is nonlinear—in order to develop a linear version of it, we introduce a set of

$$
\begin{aligned}
\min \quad & \sum_{f\in\mathcal{F}}\sum_{c\in d}\overline{dist}(c,\mathcal{FC}(c))\cdot\xi_c\cdot X_{c,f} - \lambda\sum_{c\in d}\sum_{f\in\mathcal{F}}\sum_{f'\in\mathcal{F}} Y_{c,f,f'} \\
\text{subject to} \quad & (i) \quad \sum_{f\in\mathcal{F}} X_{c,f} \geq \mu, \forall c \in \mathcal{B} \\
& (ii) \quad \sum_{c\in d} X_{c,f} \geq N_l,\ \forall f \in \mathcal{F} \\
& (iii) \quad \sum_{c\in d} \overline{dist}(c,\mathcal{FC}(c))X_{c,f} \geq T_f,\ \forall f \in \mathcal{F} \\
& (iv) \quad X_{c,f} = 0, \forall c \notin \mathcal{B}, \forall f \in \mathcal{F} \\
& (v) \quad X_{c,f} \in \{0,1\}, \forall c \in d, f \in \mathcal{F} \\
& (vi) \quad Y_{c,f,f'} \leq 2 - X_{c,f} - X_{c,f'},\ \forall c \in d, f, f' \in \mathcal{F} \\
& (vii) \quad Y_{c,f,f'} \leq X_{c,f} - X_{c,f'},\ \forall c \in d, f, f' \in \mathcal{F} \\
& (viii) \quad Y_{c,f,f'} \in \{0,1\},\ \forall c \in d, f, f' \in \mathcal{F}
\end{aligned} \tag{8}
$$

Fig. 6. Integer Linear Program Solved by WE-FORGE[I].

binary auxiliary variables $Y_{c,f,f'}$, where for each $c \in d, f, f' \in \mathcal{F}$:

$$
\begin{aligned}
& Y_{c,f,f'} \leq 2 - X_{c,f} - X_{c,f'} \\
& Y_{c,f,f'} \leq X_{c,f} + X_{c,f'} \\
& Y_{c,f,f'} \in \{0,1\}
\end{aligned} \tag{6}
$$

Now, the new objective function becomes

$$
\min \sum_{f\in\mathcal{F}}\sum_{c\in d}\overline{dist}(c,\mathcal{FC}(c))\cdot\xi_c\cdot X_{c,f} - \lambda\sum_{c\in d}\sum_{f\in\mathcal{F}}\sum_{f'\in\mathcal{F}} Y_{c,f,f'} \tag{7}
$$

As in the case of WE-FORGE[E], we have the following equivalence result:

Proposition 2. *The objective function in Equation (7) combined with Equation (6) is equivalent to the objective function in Equation (5).*

The proof is similar to the proof of Proposition 1, and we refer to the Appendix for the detailed proof. With the linearization, the implicit integration optimization WE-FORGE[I] in Equation (5) is re-written in the following form:

4.3 Overall WE-FORGE Algorithm

The overall WE-FORGE algorithm is presented in Algorithm 1. The algorithm first computes word embeddings for each concept and then computes the TFIDF of each concept in the original document d, and sorts them into bins, and then computes the feasible candidate for each selected bin. Depending on whether we are computing WE-FORGE[I] or WE-FORGE[E], two different things are done.

—If the implicit version of WE-FORGE is used, we compute the average distance from c to the set of feasible replacement concepts for c. Note that this induces a probability distribution on the set of candidate replacements for c, i.e., the members of the cluster to which c belongs. This probability assigns to each $c' \in Cluster(c)$ a probability:

$$
Prob(c') = 1 - \frac{d(c,c')}{\Sigma_{c''\in Cluster(c)} d(c,c'')}
$$

ALGORITHM 1: WE-FORGE

Input: Corpus $\mathcal{D}$, target document d, number of fake documents $|\mathcal{F}|$, parameters $\mu, \lambda, N_l, T_f, \mathcal{B}$
Output: Set of fake documents $\mathcal{F}$

1 Train word embeddings for the concepts $c \in C$;
2 Compute $\xi_c, \mathcal{FC}(c)$ for each $c \in d$;
3 **if** *Implicit Integration* **then**
4 **for** $c \in d$ **do**
5 Calculate $\overline{dist}(c, \mathcal{FC}(c))$;
6 **end**
7 Compute $X_{c,f}$, $\forall c \in d, f \in \mathcal{F}$ with Equation (8);
8 **for** $f \in \mathcal{F}$ **do**
9 Perform concept replacement according to $X_{c,f}$ from $\mathcal{FC}(c)$ stochastically;
10 **end**
11 **end**
12 **else**
13 Compute $X_{c,c',f}$, $\forall c \in d, c' \in \mathcal{FC}(c), f \in \mathcal{F}$ with Equation (4);
14 **for** $f \in \mathcal{F}$ **do**
15 Perform concept replacement according to $X_{c,c',f}$ deterministically;
16 **end**
17 **end**
18 **return** Set of fake documents $\mathcal{F}$

We then solve the linear version of the WE-FORGE optimization problem to identify the concepts to replace in any given file. We then replace the selected concepts in each file with a replacement chosen according to the distribution induced by WE-FORGE[I]. Because the replacement for a given concept in Line 9 of the algorithm is chosen according to the distribution induced by WE-FORGE[I], we note that this choice is stochastic and not deterministic.

—If the explicit version of WE-FORGE is used, then we use the ILP associated with WE-FORGE[E] to identify the (c, c', f) triples telling us to replace concept c with c' in fake file f and then replace them directly.

5 THEORETICAL RESULTS

We now study the computational complexity of our formulated optimization problems and show that solving the ILPs used by both WE-FORGE[I] and WE-FORGE[E] are NP-hard.

Theorem 1. *The WE-FORGE[I] problem described in Equation (8) is NP-Hard.*

Proof. We prove it by constructing the optimization in Equation (8) into an instance of ILP, which is known to be *NP-Hard*. The *canonical form* of an ILP is:

$$\begin{aligned} \max \quad & \mathbf{C}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & \mathbf{x} \in \mathbb{Z} \end{aligned}$$

Here, $\mathbf{C}$ is a constant weight vector of the variable $\mathbf{x}$, $\mathbf{A}$ is the coefficient matrix, and $\mathbf{b}$ is a constant vector. With this, we can transform the above formulation into the following form:

$$\max \quad \sum_{f \in \mathcal{F}} \sum_{c \in d} -\overline{dist}(c, \mathcal{FC}(c)) \cdot \xi_c \cdot X_{c,f} + \lambda \sum_{c \in d} \sum_{f \in \mathcal{F}} \sum_{f' \in \mathcal{F}} Y_{c,f,f'}$$

$$\begin{aligned}
\text{subject to} \quad & (i) \quad \sum_{f \in \mathcal{F}} -X_{c,f} \leq -\mu, \forall c \in \mathcal{B} \\
& (ii) \quad \sum_{c \in d} -X_{c,f} \leq -N_l, \ \forall f \in \mathcal{F} \\
& (iii) \quad \sum_{c \in d} -\overline{dist}(c, \mathcal{FC}(c)) X_{c,f} \leq -T_f, \ \forall f \in \mathcal{F} \\
& (iv) \quad X_{c,f} = 0, \forall c \notin \mathcal{B}, \forall f \in \mathcal{F} \\
& (v) \quad X_{c,f} \in \{0,1\}, \forall c \in d, f \in \mathcal{F} \\
& (vi) \quad X_{c,f} + X_{c,f'} + Y_{c,f,f'} \leq 2, \ \forall c \in d, f, f' \in \mathcal{F} \\
& (vii) - X_{c,f} + X_{c,f'} + Y_{c,f,f'} \leq 0, \ \forall c \in d, f, f' \in \mathcal{F} \\
& (viii) \quad Y_{c,f,f'} \in \{0,1\}, \ \forall c \in d, f, f' \in \mathcal{F}
\end{aligned}$$

Note that we can always unfold the tensor variables $X_{c,f}$ and $Y_{c,f,f'}$ into vector variables and then concatenate into a single vector variable $\mathbf{x}$. With the unfolding, we can see that constraints (i)–(iii), (vi), and (vii) correspond to $\mathbf{Ax} \leq \mathbf{b}$, while constraints (v) and (viii) correspond to $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{x} \in \mathbb{Z}$. Note that constraint (iv) is not actually used in the solving procedure, but only as a pre-solving procedure to refine the set of concepts that are eligible (in terms of TFIDF importance) of being selected, and therefore is irrelevant. From the above transformation, we have constructed the original optimization problem in Equation (8) as instance of ILP. □

The following theorem states that the same NP-hardness result also holds for WE-FORGE[E], the explicit version of the Joint Concept Replacement Problem.

Theorem 2. *The WE-FORGE[E] problem described in Equation (4) is NP-Hard.*

For the sake of brevity in the text, the proof of this Theorem is contained in the appendix. Since we have now successfully captured both variants of WE-FORGE as ILPs, existing ILP solvers could be utilized to solve our formulated optimization problems.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setting

We collected 10 Computer Science (CS) and 10 chemistry patents through Google Patents to use as the original documents that we wish to protect. For each CS document, we used WE-FORGE[I] and WE-FORGE[E] to generate 6 + 6 = 12 fake versions. In the case of CS documents, we were unable to compare with the FORGE system [5] because no ontology exists for CS as a whole. For each chemistry document, we used FORGE to generate six fake versions, and used WE-FORGE[I] and WE-FORGE[E] approaches to respectively generate three fake versions each. Therefore, we still have a total of 6+3+3=12 fake versions. We solved both ILPs used in this article with GUROBI[4] using the following parameter values.[5]

[4] https://www.gurobi.com/.

[5] Parameter values were selected based on the preliminary evaluation of a separate set of documents. In practice, these parameters can be specified by users.

Table 2. Evaluation on CS Documents

Method \ Choice	Choice 1	Choice 2	Choice 3	Total
Original	0.23	0.14	0.07	0.147
WE-FORGE[I]	**0.45**	**0.52**	**0.55**	0.506
WE-FORGE[E]	0.32	0.34	0.38	0.347

The numbers show the probability of the adversary choosing a document generated by a specific method as the true document.

Table 3. Evaluation on CS Documents by CS Graduate Students

Method \ Choice	Choice 1	Choice 2	Choice 3	Total
Original	0.31	0.18	0.03	0.173
WE-FORGE[I]	**0.42**	**0.45**	**0.56**	0.477
WE-FORGE[E]	0.27	0.37	0.41	0.350

The numbers show the probability of the adversary choosing a document generated by a specific method as the true document.

—The number of fake documents $N = |\mathcal{F}|$ is set to 4.
—The factor to tradeoff the two objective terms λ is set to 0.1.
—We split the concept into 20 bins and used concepts in bins from B_{14} to B_{19} (so that the concepts in B_1 to B_{13} are considered too unimportant to make them worth replacing while the concepts in B_{20} are the most important concepts in the document and replacing them increases the likelihood of the adversary detecting fakes).
—We also require that each concept in the selected bins must be replaced at least $\mu = 2$ times across all generated fake documents.
—The number of concepts replaced in a document is at least $N_l = 4$.
—The normalized threshold for concept distance T_f is set to 0.2.

6.2 Human Evaluation Using Amazon Mechanical Turk

We designed two sets of human evaluations for the CS and chemistry documents, respectively. In each assessment, we invited 10 participants from Amazon Mechanical Turk who have a "Master" level recognition from the M-Turk platform. In each of the 10 CS/Chemistry tasks, the participants were asked to read the 13 documents (1 original plus 12 fake), and select the document they felt was the most, the second most, and the third most likely to be the real one.

We provided a financial incentive to participants to make a strong effort to identify the real documents. In each task, participants received a score of 3, 2, and 1, respectively, if they correctly identified the genuine document in their first, second, and third top choices. Participants were told that the three participants with the three highest scores would receive an extra payment.

6.3 Experimental Results

6.3.1 Believability of Generated Documents. This experiment evaluated the believability of the generated documents and WE-FORGE's success in deceiving an adversary whose task it is to identify the fakes. Tables 2 and 4 show the believability results of the CS documents, while Tables 6 and 7 show the believability results of the chemistry documents. In all of Tables 2–7, the first column describes the type of documents generated by the following five methods:

Table 4. Evaluation on CS Documents

Method \ Task	1	2	3	4	5	6	7	8	9	10	Total
Original	0.007	0.023	0.010	0.010	0.007	0.013	0.017	0.017	0.023	0.020	0.147
WE-FORGE[I]	0.063	0.037	0.047	0.057	0.057	0.053	0.050	0.043	0.047	0.053	0.506
WE-FORGE[E]	0.030	0.040	0.043	0.033	0.037	0.033	0.033	0.027	0.030	0.027	0.347

The numbers show the probability of the adversary choosing a document generated by a specific method as the true document.

- **Original.** This refers to the original (real) document. As described in Section 6.2, only 1 out of the 13 documents is real.
- **WE-FORGE[I].** This means the documents were generated using WE-FORGE[I]. Note that for CS documents, 6 documents out of the 13 are generated by this method, while for chemistry documents, 3 out of 13 are generated by this method.
- **WE-FORGE[E].** This means the documents were generated using WE-FORGE[E]. Similarly, 6 out of 13 documents in CS and 3 out of 13 documents in chemistry are generated by this method.
- **WE-FORGE.** This is a combination of WE-FORGE[I] and WE-FORGE[E]. Column WE-FORGE in Tables 6 and 7 is the sum of Columns WE-FORGE[I] and WE-FORGE[E].
- **FORGE.** The fake document generation method used in Ref. [5], which uses an ontology to generate fake documents. We had no ontology for CS documents, and so it was not applied to CS documents. For chemistry documents, 6 out of 13 documents are generated with FORGE.

The numbers in each column of Tables 2 and 6 refer to the probability that the original document was picked vs. documents generated by WE-FORGE[I] and WE-FORGE[E]. For instance, the Choice 1 column in Table 2 says that 23% of the first choice made by our subjects correctly identified the original document, 45% identified a fake generated by WE-FORGE[I], and 32% identified a fake generated by WE-FORGE[E]. The final column aggregates the numbers from the previous columns. The best results at deception are highlighted in boldface. We see that WE-FORGE[I] has the best result in both the CS and Chemistry datasets.

The numbers in Tables 4 and 7 refer to the number of times that the documents generated by different methods are selected by human participants in different tasks. The last column is a summation of the previous columns. In these two tables, we do not distinguish between different choices. That is, we add 1 to the count as long as a document generated by a certain method is selected as any one of the three choices.

Results on CS Documents. We see that (1) both WE-FORGE[I] and WE-FORGE[E] are able to generate fake documents that are highly believable; (2) WE-FORGE[I] is, in general, better than WE-FORGE[E] at deception. Our conjecture for the second observation is that the word replacement generation in WE-FORGE[I] is stochastic, while it is deterministic in WE-FORGE[E] (therefore, more likely to be reverse engineered to the real document once a replacement is identified).

Table 4 shows the selections for each task, where the top row indicates the task number. In this table, we do not differentiate between first, second, or third top choice, i.e., once a document is selected as a genuine document by a participant as any one of the three choices, we add 1 to the count. We can see that (1) both WE-FORGE[I] and WE-FORGE[E] consistently generate highly believable fake documents for each task; (2) WE-FORGE[I] is better than WE-FORGE[E] in most but not all tasks.

Table 5. Evaluation on CS Documents by CS Graduate Students

Method \ Task	1	2	3	4	5	6	7	8	9	10	Total
Original	0.010	0.023	0.017	0.010	0.013	0.017	0.023	0.010	0.030	0.020	0.173
WE-FORGE[I]	0.047	0.040	0.036	0.070	0.047	0.046	0.050	0.052	0.040	0.050	0.477
WE-FORGE[E]	0.043	0.037	0.047	0.020	0.040	0.037	0.027	0.038	0.030	0.030	0.350

The numbers show the probability of the adversary choosing a document generated by a specific method as the true document.

Table 6. Evaluation on Chemistry Documents

Method \ Choice	Choice 1	Choice 2	Choice 3	Total
Original	0.040	0.030	0.043	0.113
WE-FORGE[I]	0.107	0.107	0.113	0.327
WE-FORGE[E]	0.070	0.107	0.083	0.260
WE-FORGE	0.177	0.213	0.197	0.587
FORGE	0.117	0.090	0.093	0.300

The numbers show the probability of the adversary choosing a document generated by a specific method as the true document.

Table 7. Evaluation on Chemistry Documents

Method \ Task	1	2	3	4	5	6	7	8	9	10	Total
Original	0.003	0.020	0.017	0.010	0.010	0.010	0.010	0.010	0.013	0.010	0.113
WE-FORGE[I]	0.030	0.037	0.023	0.030	0.033	0.043	0.020	0.037	0.033	0.040	0.327
WE-FORGE[E]	0.020	0.023	0.020	0.027	0.037	0.023	0.030	0.023	0.033	0.023	0.260
WE-FORGE	0.050	0.060	0.043	0.057	0.070	0.067	0.050	0.060	0.067	0.063	0.587
FORGE	0.047	0.020	0.040	0.033	0.020	0.023	0.040	0.030	0.020	0.027	0.300

The numbers show the probability of the adversary choosing a document generated by a specific method as the true document.

In order to ensure that the human participants have solid domain knowledge about the documents, we have also invited 10 Computer Science MS and PhD students to participate in the experiments. The results are shown in Tables 3 and 5. We can see that similar results can be observed with CS graduate students.

Results on Chemistry Documents. Tables 6 and 7 show similar believability results for the chemistry documents. Note that since we have an ontology for chemistry documents from Ref. [5], we can compare WE-FORGE with the FORGE method here.

The results show that in the case of our chemistry documents, the results obtained are similar to those with CS documents. However, we have two additional observations: (1) the deceptiveness of documents generated by WE-FORGE is higher in chemistry documents, (2) WE-FORGE is much better than FORGE, in general, and is better than FORGE in 9 of the 10 tasks. This suggests that WE-FORGE[I] both involves less manual labor (i.e., no need to painstakingly create an ontology) and achieves a higher degree of deception than FORGE.

Table 8. Runtime (in seconds) Comparison between WE-FORGE[I] and WE-FORGE[E]

Patent		Chemistry 1	Chemistry 2	Chemistry 3	Chemistry 4	Chemistry 5	CS 1	CS 2	CS 3	CS 4	CS 5
Default	Explicit	0.469	0.147	0.166	0.298	0.166	0.951	1.007	1.615	0.533	1.081
Setting	Implicit	0.071	0.059	0.072	0.0816	0.071	0.092	0.079	0.119	0.085	0.848
$N_l = 6$	Explicit	0.455	0.144	0.162	0.293	0.162	0.683	0.973	1.607	0.451	0.993
	Implicit	0.068	0.054	0.061	0.076	0.065	0.089	0.075	0.096	0.078	0.086
$\mu = 4$	Explicit	0.453	0.141	0.158	0.284	0.158	0.667	0.956	1.572	0.518	1.039
	Implicit	0.043	0.039	0.040	0.042	0.040	0.045	0.043	0.048	0.044	0.044
$T_f = 0.4$	Explicit	0.453	0.143	0.160	0.287	0.161	0.673	0.962	1.575	0.519	1.042
	Implicit	0.078	0.058	0.061	0.081	0.066	0.091	0.077	0.099	0.088	0.075
$\mathcal{B} =$	Explicit	0.494	0.189	0.248	0.499	0.232	1.674	1.279	2.456	1.595	1.134
$B_{10} - B_{19}$	Implicit	0.076	0.069	0.077	0.079	0.071	0.102	0.102	0.149	0.102	0.086

The default hyperparameters are set as: $N = |\mathcal{F}| = 4$, $\mathcal{B}$ = bins 14–19, $N_l = 4$, $\mu = 2$, $T_f = 0.2$, only the modified parameters are shown for each row.

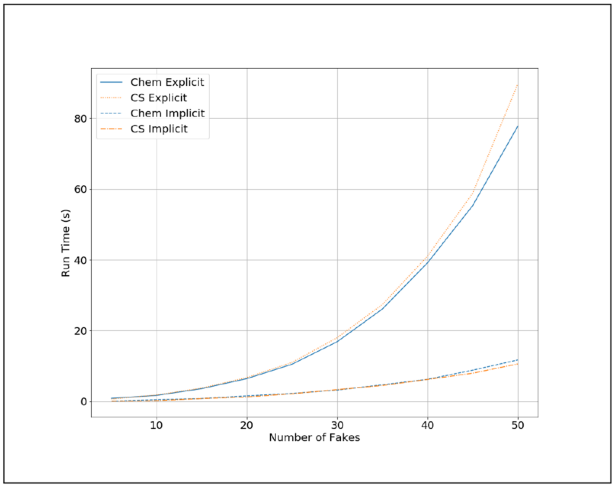

Fig. 7. Comparative Runtimes as the number of fake documents is increased from 5 to 50 in Steps of 10.

6.3.2 Runtime. We evaluated the relative runtime of WE-FORGE[I] and WE-FORGE[E] in five settings. Our default setting is specified in Section 6.1, and only one hyperparameter is changed as shown in each row.

Runtime under Different Settings. Table 8 shows the runtimes for 10 randomly chosen documents, five each in Chemistry and CS. We see that WE-FORGE[I] is faster than WE-FORGE[E] in all cases.

Runtime as number of fake documents is increased. We also assessed how WE-FORGE[E] and WE-FORGE[I]'s runtime changes as the number of fake files to be generated increase from 5–50 files in steps of 5. We show the results under the default setting (except for that $|\mathcal{F}|$ varies over the set $\{5, 10, 15, \ldots, 45, 50\}$). The results are shown in Figure 7, which reports the average time taken across all the 20 documents (10 each in CS and Chemistry). wWe can see that WE-FORGE[I] is much faster than WE-FORGE[E], increasing almost linearly, while WE-FORGE[E] has an exponential growth in runtime.

6.3.3 Time Taken by Human Participants. Table 9 shows the average time it took participants to complete a task. On average, each task takes 11.89, 19.71, and 14.89 minutes, respectively, for CS

Table 9. Time Taken (in minutes) by Human Subjects to Complete Each Task

Task	1	2	3	4	5	6	7	8	9	10	Avg
CS Docs	16.8	10.4	11.5	11.3	11.1	12.4	11.6	10.9	13.0	10.8	11.98
CS Docs graduate students	27.3	20.2	19.4	15.5	24.7	18.5	22.0	20.0	15.0	14.5	19.71
Chemistry Docs	18.7	17.0	14.6	14.7	15.1	13.9	11.2	15.5	13.6	14.6	14.89

Table 10. Confidence Level of Subjects in CS and Chemistry on a 1 (Very Low) to 5 (Very High) Scale

Task	1	2	3	4	5	6	7	8	9	10	Avg
Computer Science Docs	3.1	3.4	3.0	3.1	3.0	3.4	3.2	2.9	2.9	2.9	3.09
Chemistry Docs	3.0	3.2	3.2	2.9	2.6	2.7	2.6	2.5	2.4	2.5	2.76

Table 11. Average Jaccard Similarity Values of Generated Fake Documents with and without Diversity Regularization Terms, with Standard Deviations

Method	WE-FORGE[E]		WE-FORGE[I]	
Diversity term	without	with	without	with
Jaccard similarity	0.821 ± 0.011	0.179 ± 0.080	0.864 ± 0.021	0.5637 ± 0.147

(by Mturk), CS (by CS graduate students), and chemistry documents, with standard deviations of 1.76, 3.40, and 1.89 minutes. In general, this suggests that the subjects took one minute on average per document that they considered.

6.3.4 Confidence Level. Table 10 shows the average confidence level from the 10 participants for each task. The confidence level ranges from 1 to 5, where 1 indicates a complete guess, and 5 means they were 100% confident. On average, the confidence levels for the CS and chemistry documents are 3.09 and 2.76, with variances of 0.18 and 0.28. This suggests that participants only had moderate levels of confidence in the accuracy of their guesses.

6.3.5 Diversity of Generated Fake Documents. To evaluate the diversity of the generated fake documents, we conduct an ablation study by comparing the similarity of generated documents with and without the diversity regularization term (i.e., the second term in the objective functions in Equations (1) and (5)). More specifically, we do a pairwise comparison of the fake documents in $\mathcal{F}$. Suppose Jaccard(f', f'') denotes the Jaccard similarity of two fake documents (f', f'') in $\mathcal{F}$. We define the average Jaccard similarity of all the pairs of fake documents in $\mathcal{F}$ as

$$\text{Jaccard}(\mathcal{F}) = \frac{2}{|\mathcal{F}||\mathcal{F}-1|} \sum_{f' \in \mathcal{F}} \sum_{f'' \in \mathcal{F} \& f'' \neq f'} \text{Jaccard}(f', f'').$$

Note that the total number of such pairs is $\frac{|\mathcal{F}||\mathcal{F}-1|}{2}$.

The results are shown in Table 11. We can see that for both WE-FORGE[I] and WE-FORGE[E], the Jaccard similarity of the generated fake documents with the diversity regularization term is much smaller than without it. This indicates that the diversity of the generated documents is indeed increased with the diversity regularization—the regularization term in the objective function ensures that the Jaccard similarity of the generated fake documents is smaller.

7 TWO OTHER ISSUES

In this section, we discuss two other issues.

The Role of the Author. In the WE-FORGE system, an appropriate user (e.g., the author of a document that needs to be protected from IP theft) is charged with using WE-FORGE to generate the fakes. When this user runs WE-FORGE on a document d, it generates candidate concepts c for replacement along with a selected replacement c' for each candidate concept c. In addition, it lists the entire contents $Cand(c)$ of the cluster to which c belongs. The user can choose to go with the suggested replacement c' or select another replacement c'' from $Cand(c)$. He may also make up his own custom replacement concept $c^\star$.

Separating Fakes from the Real Documents. The reader may legitimately wonder: if an enterprise's network has $|\mathcal{F}|$ fake documents associated with each real document (presumably the important ones), how will a legitimate user be able to tell which document is real and which one is fake? The FORGE system [5] proposed a solution to this problem by embedding a message authenticating code within each document (real or fake). The user's private key will enable him to identify which of $|\mathcal{F}| + 1$ versions of a document is the real one. We do not go into details here as this problem has already been solved in Ref. [5], and WE-FORGE merely uses the same method.

8 CONCLUSION

The problem of deterring IP theft is critical to the economies of countries developing advanced technology. Large investments are made by innovative companies—yet, a relatively cheap cyber attack can enable an attacker to steal hundreds of millions of dollars of new technology. Moreover, according to Symantec researchers, victims discover that their enterprise has been compromised by a zero-day attack only after 312 days on average [1], giving attackers a huge window of time during which they can steal valuable IP. The goal of this article is to deter such attackers by imposing costs and increasing uncertainty for them. The recent FORGE system [5] was an important first effort in this direction.

The WE-FORGE system presented here has four major advantages over FORGE: (i) because it does not need an ontology, it is far more widely applicable than FORGE specially for domains where good ontologies are not available; (ii) it selects concept-replacement pairs rather than concepts first and then replacements, thus avoiding making a sub-optimal choice; (iii) it ensures that the set of generated fake documents are diverse in the sense that there is a decent range of difference among them unlike FORGE, which could not guarantee this. (iv) The WE-FORGE[I] algorithm ensures that replacements are chosen stochastically, thus reducing the chances that an adversary can easily reverse engineer the replacements. In particular, our experiments show that WE-FORGE achieves a higher rate of deception compared to FORGE *without requiring that an ontology exist or be developed.* Moreover, we show on a realistic set of 20 patents that WE-FORGE[I] runs in a reasonable amount of time—between 0 and 12 second even if 50 fake documents are to be generated. This suggests that WE-FORGE provides superior performance when compared to FORGE. Furthermore, WE-FORGE[I] beats WE-FORGE[E] in deceiving adversaries—this is likely because WE-FORGE[I] makes stochastic choices in replacements while WE-FORGE[E] is more deterministic (and, hence, perhaps easier for an adversary to guess).

Of course, there are many important directions for future research. FORGE and WE-FORGE only modify the textual part of a document. But a document can contain diverse types of interlinked entities such as figures, flowcharts, and tables. We need to be able to ensure that changes in the text and consistently reflected across these kinds of entities. This would be a major next step which we plan to study.

REFERENCES

[1] Leyla Bilge and Tudor Dumitraş. 2012. Before we knew it: An empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. 833–844.

[2] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. "O'Reilly Media, Inc."

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[4] Brian M. Bowen, Shlomo Hershkop, Angelos D. Keromytis, and Salvatore J. Stolfo. 2009. Baiting inside attackers using decoy documents. In *Proceedings of the International Conference on Security and Privacy in Communication Systems*. Springer, 51–70.

[5] Tanmoy Chakraborty, Sushil Jajodia, Jonathan Katz, Antonio Picariello, Giancarlo Sperli, and V.S. Subrahmanian. 2019. FORGE: A fake online repository generation engine for cyber deception. *IEEE Transactions on Dependable and Secure Computing*.

[6] Hans Christian, Mikhael Pramodana Agus, and Derwin Suhartono. 2016. Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF). *ComTech: Computer, Mathematics and Engineering Applications* 7, 4 (2016), 285–294.

[7] Catalin Cimpanu. 2020. FBI is investigating more than 1,000 cases of Chinese theft of US technology. Retrieved from https://www.zdnet.com/article/fbi-is-investigating-more-than-1000-cases-of-chinese-theft-of-us-technology/.

[8] Paul K. Huth. 1999. Deterrence and International Conflict: Empirical findings and theoretical debates. *Annual Review of Political Science* 2, 1 (1999), 25–48. DOI: https://doi.org/10.1146/annurev.polisci.2.1.25

[9] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2020. *Mining of Massive Data Sets*. Cambridge University Press.

[10] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. Oakland, CA, 281–297.

[11] Francois Mathey, Francois Mercier, Michel Spagnol, Frederic Robin, and Virginie Mouries. 2003. 6, 6′-bis-(1-phosphanorbornadiene) diphosphines, their preparation and their uses. US Patent 6,521,795.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[13] Younghee Park and Salvatore J. Stolfo. 2012. Software decoys for insider threat. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. 93–94.

[14] Eric Rosenbaum. 2019. 1 in 5 corporations say China has stolen their IP within the last year: CNBC CFO survey. Retrieved from https://www.cnbc.com/2019/02/28/1-in-5-companies-say-china-stole-their-ip-within-the-last-year-cnbc.html.

[15] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20 (1987), 53–65.

[16] Thomas C. Schelling. 2008. Arms and influence. In *Strategic Studies*. Routledge, 96–114.

[17] Elena Paslaru Bontas Simperl, Christoph Tempich, and York Sure. 2006. Ontocom: A cost estimation model for ontology engineering. In *Proceedings of the International Semantic Web Conference*. Springer, 625–639.

[18] Venkatramana S. Subrahmanian and Diego Reforgiato. 2008. AVA: Adjective-verb-adverb combinations for sentiment analysis. *IEEE Intelligent Systems* 23, 4 (2008), 43–50.

[19] Jonathan Voris, Nathaniel Boggs, and Salvatore J. Stolfo. 2012. Lost in translation: Improving decoy documents via automated translation. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy Workshops*. IEEE, 129–133.

[20] Lei Wang, Chenglong Li, QingFeng Tan, and XueBin Wang. 2013. Generation and distribution of decoy document system. In *Proceedings of the International Conference on Trustworthy Computing and Services*. Springer, 123–129.

[21] Jonathan White and Dale Thompson. 2006. Using synthetic decoys to digitally watermark personally-identifying data and to promote data security. In *Security and Management*. Citeseer, 91–99.

[22] Ben Whitham. 2013. Automating the generation of fake documents to detect network intruders. *International Journal of Cyber-Security and Digital Forensics* 2, 1 (2013), 103.

[23] Ben Whitham. 2014. Design requirements for generating deceptive content to protect document repositories. In *Proceedings in the 15th Australian Information Warfare Conference, Perth, Australia*.

[24] Jim Yuill, Mike Zappe, Dorothy Denning, and Fred Feer. 2004. Honeyfiles: Deceptive files for intrusion detection. In *Proceedings from the 5th Annual IEEE SMC Information Assurance Workshop, 2004*. IEEE, 116–122.

Received April 2020; revised June 2020; accepted August 2020