Generating Realistic Fake Equations in Order to Reduce Intellectual Property Theft

Yanhai Xiong*, Giridhar Kaushik Ramachandran†, Rajesh Ganesan†, Sushil Jajodia† and V.S. Subrahmanian*
* Dartmouth College, Hanover, NH 03755, USA
† George Mason University, Fairfax, VA 22030, USA

Abstract—According to Symantec, there average gap from the time a company is compromised by a zero-day attack to the time the vulnerability is discovered is 312 days. This leaves an adversary with a lot of time to exfiltrate corporate IP. Recent work has suggested automatically generating multiple fake versions of a document to impose costs on the attacker who needs to correctly identify the original document from a set of mostly fake documents. But in the real world, documents contain many diverse components. In this paper, we focus on technical documents that often contain equations. We present FEE (Fake Equation Engine), a framework to generate fake equations in such documents. FEE tries to preserve multiple aspects of a given equation when generating a fake. Moreover, FEE is very general and applies to diverse equational forms including polynomial equations, differential equations, transcendental equations, and more. FEE iteratively solves a complex, changing optimization problem inside it. We also present FEE-FAST, a fast approximate algorithm to solve the optimization problem within FEE. Using a panel of human subjects, we show that FEE achieves a high rate in deceiving sophisticated subjects.

Index Terms—cybersecurity, intellectual property theft, deception

1 Introduction

According to research from Symantec [4], on average, there is a gap of 312 days from the time a company is compromised by a zero-day attack to the time the vulnerability is detected. During this time, the attacker can easily steal a huge amount of intellectual property.

As a compromised enterprise doesn't even know they have been compromised, there is a need for techniques that automatically penalize the attacker by: delaying him, increasing his level of frustration, adding financial costs, and increasing his uncertainty. Recent work [22], [14], [18], [5], [6] has suggested that for any given original document d, we generate N fake versions of d such that it is hard for the attacker to separate the original document from the fakes. The fakes should be "similar enough" to the original to make them credible to experts in the field, yet "dissimilar enough" to make them likely to be wrong. The attacker will need to spend time to identify the real one — and even after making a decision, will be unsure about whether he was right or wrong. Simply put, generating fake documents deters attackers by imposing delays, financial costs, frustration, and uncertainty on the attackers.

Past work in this relatively new area focuses on the textual part ([22], [14], [5]) or tabular data ([18], [6]) in a document. However, technical documents have many components: diagrams, images, equations, tables, and more. Equations are at the very heart of technical documents because a small change in an equation can completely alter its meaning and/or render it incorrectly. And finding errors in complex equations is not always an easy task, especially if the errors are subtle.

Manuscript received; revised.

In this paper, we focus on taking an equation that might occur in a technical document and generating k fake versions of it so that the resulting fake equations are "similar enough" to the original equation to be credible, but sufficiently "dissimilar" to likely be wrong. However, equations are not just pieces of syntax. An equation has a semantic meaning. The equation $y = 2x + 4$ has a physical meaning — it denotes a line in a 2-dimensional space with a slope of 2 and a y-intercept of 4. When we talk about one equation being "similar" or "dissimilar" to another, this semantic meaning should be taken into account. So should the form of the equation (e.g. linear equation vs. differential equation), universal truths (e.g. weight must exceed 0), and consistency with other equations within the same document. Moreover, the space of possible equations is enormous.

All of these factors make the generation of fake equations a very challenging task. In this paper, we propose a novel system called FEE (Fake Equation Engine) with the following characteristics:

1) FEE takes as input, a grammar which can capture many different types of equations. FEE then modifies equations by applying some edit operators, each with a given cost.
2) However, FEE must generate fake equations that satisfy various desired constraints — which at the same time conflict with each other. We therefore write FEE down as an iterative algorithm which invokes a very non-traditional optimization problem called FEE[OPT] within it. This optimization problem cannot be solved by a standard optimization problem. Moreover, FEE[OPT] changes from one iteration of FEE to the next and is very challenging

to solve.

3) We therefore develop a separate algorithm called FEE-FAST that approximately solves this optimization problem.

In all, FEE generates a desired set of k fake equations for any given real equation. We tested FEE out on a panel of 50 subjects on Amazon Mechanical Turk, each of who is in the US and has a Master's degree or higher. The subjects were given 20 tasks, each of which involved identifying the correct equation from a set of 11 equations (10 fake, one real). We defined the notion of *deception factor* and show that FEE achieves a high deception factor: most subjects were effectively deceived by FEE even though a number of conditions favored the subjects' in their quest to find the right equation.

The remainder of this paper is organized as follows. Section 2 presents a quick overview of related work. Section 3 presents the architecture of the overall FEE framework. After this, Section 4 presents the form in which equations are considered by FEE through context free grammars and show that this syntax is enough to represent polynomial, differential, and transcendental equations. Section 5 shows the main contributions of the paper including the overall FEE algorithm, the FEE[OPT] optimization problem, and the FEE-FAST algorithm to solve FEE[OPT]. We then present our experimental results in Section 6 after which we present conclusions and future work.

2 Related Work

The use of deception in warfare goes back many centuries [16], [9]. In the context of cybersecurity, deception has primarily been used by the attacker — for instance, phishing attacks try to deceive a victim into downloading malware or otherwise being compromised.

The use of cyber-deception [13] for defensive purposes is newer. Issues such as piracy on the Internet (e.g. of videos) [24], audio (e.g. music recordings) [15] and software code [19] have led to the creation of a host of watermarking and steganography techniques so that legitimate owners of music or videos can show clear evidence of piracy. [20] provides an excellent survey of methods to identify data leakage from organizations using such methods.

One class of methods to protect technical documents involve the creation of a "decoy" document [25], [23], [21]. [21] proposes Canary files. If the content of a canary file is accessed or copied or deleted, then the system administrator is immediately notified about the access. [23] mentions two methods to generate honeyfiles with different levels of permitted interaction. [25] uses honeyfiles to send alarms when intrusions are detected. Such research on honeyfiles focus on generating alarms when fake documents are touched.

In general, generating decoy technical documents involve handling the fact that technical documents contain diverse many constituent parts such as text, tables, graphs, equations and flow-charts etc. [5] develops the FORGE (Fake Online Repository Generation Engine) system in which fake versions of the textual part of a document are automatically generated using a mix of three methods: natural language processing, multi-layered graph "meta-centrality" measures, and optimization. [22] use word transposition and substitution based on parts of speech tagging and pre-collected n-grams to generate fake text. [14] focuses on increasing comprehension burden for attackers through shuffling, deletion and addition of concepts. [18], [6] provide methods for synthesizing fake tables for large data using Generative Adversarial networks.

However, to date, there has been no work that we are aware of that specifically tries to generate fake equations that may occur within a technical document. An equation is a model of some underlying phenomenon: for instance, the famous $e = mc^2$ is really a model that captures the relationship between energy (e), mass (m) and the speed of light (c). Equations are typically derived in one of two ways. The equations could be derived from a body of data which already exists (e.g. regression equations [3]). Alternatively, the equations may constitute a theoretical model — and experiments to gather data to validate the theoretical method may be *subsequently* generated. This happens frequently in physics. In this paper, we develop methods to deceive attackers when they have access to the actual document containing an equation but the paper itself doesn't contain the extensive data needed to support the equations in the document (if in fact that data even exists). To the best of our knowledge, there is no work on generating fake equations under these conditions which are the widespread as far as technical documents are concerned.

There are also efforts that focus on generating synthetic *structured* datasets such as relational databases [6] and specialized data sets such as network traces [17]. Those are important efforts that are orthogonal to ours.

Unlike the above efforts, we focus on manipulating equations within the technical document. The FEE framework we propose is based on context-free grammars [12] and can automatically generate fake equations that are "similar enough" to the original equation to be realistic, yet "dissimilar enough" to the original equation to likely be wrong.

3 FEE Architecture

Figure 1 shows the architecture of the FEE framework. The system takes as input, an original document with a real equation. The goal is to generate a number of fake documents, each with a fake version of this equation.

FEE contains a suite of context free grammars (CFGs) that each try to parse the equation. Currently, we have developed CFGs for polynomial, differential, and transcendental equations. Each of the CFGs tries to parse the equation e — as long as one of the CFGs accepts the equation, we can generate fake versions of it.[1] The creation of such CFGs is easy and many example CFGs for different equational forms already exist [1]. A rich body of work exists on recognizing mathematical expressions from handwritten documents [10] as well as from printed documents [2]. We therefore do not delve deeply into this part of FEE in this paper.

1. If no CFG in the suite accepts an equation, then this means that the equation is of a form different from those that we have considered in the library. In such cases a new CFG must be created.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3038132, IEEE Transactions on Dependable and Secure Computing

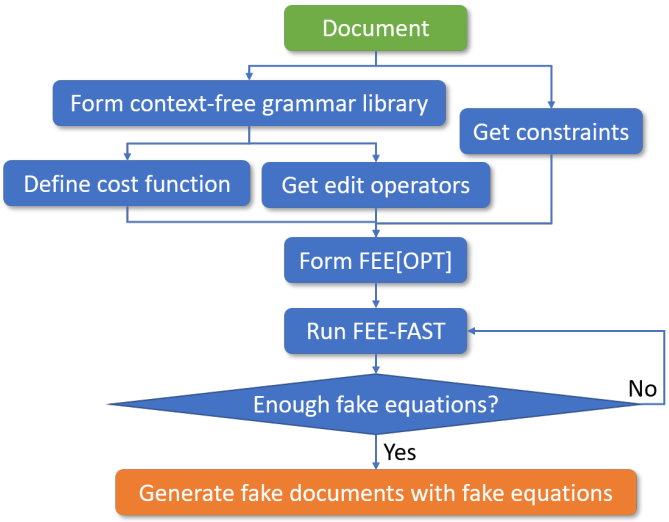


Fig. 1. FEE architecture

Once an equation is parsed into its constituent parts by one of the parsers in FEE, the FEE algorithm uses a set of edit operators and a cost function to pose an optimization problem that we call FEE[OPT]. This is a highly non-traditional optimization problem which we solve using the FEE-FAST algorithm. The main FEE algorithm involves FEE-FAST iteratively — but in each iteration, a modified version of the optimization problem is created and solved. The algorithm finally outputs a set of k fake equations. The novel contributions of this paper include the overall FEE architecture as well as the FEE algorithm consisting of both FEE[OPT] and FEE-FAST, together with the experiments documenting their efficacy.

4 Representing and Manipulating Equations via CFGs

Though all readers are familiar with the intuitive concept of an equation, we need to formally define the types of equations manipulated by the FEE framework. We use context-free grammars or CFGs [12] in order to express equations. We recall that a CFG $\mathcal{G} = (V_G, T_G, R_G, S_G)$ consists of 4 parts where V_G is a set whose elements are called *variable* symbols, T_G is a set disjoint from V_G whose elements are called terminal symbols, R_G is a finite set of "production rules" (defined below) and $S_G \in V_G$ is a distinguished variable called the "start" variable for the grammar G.

A production rule is an expression of the form $X \to Y$ where $X \in V_G$ is a variable and Y is a string (possibly empty) constructed from the set $(V_G \cup T_G)^\star$.[2]

Throughout this paper, we assume that any given equation is accepted by a grammar $\mathcal{G}$ in FEE's repository of grammars. We use $\mathcal{G}^\star$ to denote the set of all strings accepted by grammar $\mathcal{G}$.

Throughout this paper, we will use three grammars $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ as running examples in order to illustrate the definitions, concepts, and algorithms in the paper.

2. Given a set of production rules, it is common to denote the set of all strings including the empty string generated by a set Σ of symbols by $\Sigma^\star$.

Example 1 (Polynomial Equation Grammar $\mathcal{G}_1$). Suppose V_G is a set of variables, T_G includes the set consisting of some finite decimal numbers in set $\mathbb{R}$ as well as the set $\mathbb{X} = \{x_1, \ldots, x_n\}$ for some n, and some fixed set $\mathbb{O} = \mathbb{O}^1 \cup \mathbb{O}^2$ of operations, where $\mathbb{O}^1 = \{()\}$ is a set of unary operations and $\mathbb{O}^2 = \{+, -, \times, \div\}$ is a set of binary operations. In this case, our CFG might contain the following production rules:

$$\begin{aligned} S_G &\to V = V \\ V &\to t \ \forall t \in \mathbb{R} \cup \mathbb{X} \\ V &\to V + V \\ V &\to V - V \\ V &\to V \times V \\ V &\to V \div V \\ V &\to (V) \end{aligned}$$

This grammar accepts all polynomial equations containing the sole operators $+, -, \times, \div, ()$. The set of equations accepted by this grammar therefore includes: $x_1 = 3 \times x_2 - 4 \times x_3 + 5 \times x_4 \times x_4$ and $x_1 = 2 \div (x_2 + 5)$ etc. Of course, this grammar can be easily modified to include other types of operators (e.g. exponentiation).

We also define *term*s that occur in any string accepted by the grammar $\mathcal{G}$ as follows.

Definition 1 (Term). A *term* $\hat{s}$ in a string s is a **sub-string** of s (we use $\sqsubseteq$ to denote the subterm relationship, i.e. $\hat{s} \sqsubseteq s$) with length greater than 1 such that: (i) $\hat{s}$ is accepted by the grammar and (ii) $\hat{s}$ contains at least one element from $\mathbb{X}$. Therefore, when $\hat{s} \sqsubseteq s$, $len(\hat{s}) > 1$ and $\hat{s} \in \mathcal{G}^\star$, we have

- if $\exists t \in \mathbb{X}$ such that $t \sqsubset \hat{s}$, then $\hat{s}$ is a *term*;
- if $\hat{s}$ is a *term*, then $\forall O \in \mathbb{O}^1$, any $O(\hat{s}) \sqsubseteq s$ is a *term*;
- if $\hat{s}$ is a *term* and V is a variable (as defined in production rules of Example 1), then $\forall O \in \mathbb{O}^2$, any $O(\hat{s}, V) \sqsubseteq s$ is a *term* and the same for $O(V, \hat{s}) \sqsubseteq s$.

Thus, for the sample equation $x_1 = 3 \times x_2 - 4 \times x_3 + 5 \times x_4 \times x_4$ which is accepted by the grammar $\mathcal{G}_1$, both $3 \times x_2$ and $3 \times x_2 - 4 \times x_3$ are *term*s, but $3 \times x_2 - 4\times$ is not a valid term.

Example 2 (Differential Equation Grammar $\mathcal{G}_2$). Suppose V_G is a set of variables and T_G is the union of the sets $\mathbb{R}, \mathbb{X}, \mathbb{O}$, where $\mathbb{R}$ is a set of finite decimal numbers, $\mathbb{X} = \{x_1, \ldots, x_n\}$ for some n and $\mathbb{O} = \mathbb{O}^1 \cup \mathbb{O}^2 = \{+, -, \times, \div, \partial/\partial, ()\}$ is some fixed set of operations. In this case, the CFG includes following production rules:

$$\begin{aligned} S_G &\to V = V \\ V &\to t \ \forall t \in \mathbb{R} \cup \mathbb{X} \\ V &\to \partial V / \partial V \\ V &\to V + V \\ V &\to V - V \\ V &\to V \times V \\ V &\to V \div V \\ V &\to (V) \end{aligned}$$

This grammar accepts all polynomial differential equations that involving the simple operators $+, -, \times, \div, ()$. The set of equations accepted by this grammar includes $\partial(y_1 + x_1 \times x_1)/\partial x_1 = 3 \times x_2$ and $\partial y_1/\partial(x_1 \times x_1 + 2) = \partial(\partial y_2/\partial x_1)/\partial x_2 + 3 \times x_1 - y_1$ etc. In the case of the sample equation $\partial(y_1 + x_1 \times x_1)/\partial x_1 = 3 \times x_2$, $y_1 + x_1 \times x_1$, $\partial(y_1 + x_1 \times x_1)/\partial x_1$ and $3 \times x_2$ are all valid terms.

Example 3 (Transcendental Equation Grammar $\mathcal{G}_3$). Suppose V_G is a set of variables, T_G is the union set of finite decimal numbers $\mathbb{R}$, $\mathbb{X} = \{x_1, \ldots, x_n\}$ for some n, a set of unary and binary transcendental functions $\mathbb{F} = \{\sin, \cos, \exp, \log_e, \log_n, \hat{}\}$ and some fixed set of operations $\mathbb{O} = \mathbb{O}^1 \cup \mathbb{O}^2 = \{+, -, \times, \div, ()\}$. In this case, the CFG includes the following production rules:

$$
\begin{array}{rcl}
S_G & \rightarrow & V = V \\
V & \rightarrow & t \ \forall t \in \mathbb{R} \cup \mathbb{X} \\
V & \rightarrow & fV \ \forall f \in \mathbb{F}^1 \\
V & \rightarrow & VfV \ \forall f \in \mathbb{F}^2 \\
V & \rightarrow & V + V \\
V & \rightarrow & V - V \\
V & \rightarrow & V \times V \\
V & \rightarrow & V \div V \\
V & \rightarrow & (V)
\end{array}
$$

Note that $\mathbb{F} = \mathbb{F}^1 \cup \mathbb{F}^2$, $\mathbb{F}^1$ is the set of unary transcendental functions ($\sin, \cos, \exp$ etc.), while $\mathbb{F}^2$ is for binary transcendental functions (e.g., $\{\hat{}\}$). This grammar accepts all transcendental equations that involve transcendental functions in the set $\mathbb{F}$. The set of equations accepted by this grammar includes $\sin x_1 = \exp(2 + x_3\hat{}(x_2 - 1))$ and $x_1 = \log_e(x_2 \times x_2 + 1) \div 2 - x_3 + 1$ etc. For the sample equation $\sin x_1 = \exp(2 + x_3\hat{}(x_2 - 1))$, some instances of *term*s are $x_3\hat{}(x_2 - 1)$ and $\sin x_1$.

It is important to note that FEE currently contains only these three grammars. However, grammars for other types of equational forms can easily be added in a production version of such a system. We now introduce edit operations that modify equations.

Definition 2 (Edit Operator). An *edit operator* ρ is a mapping from $\mathcal{G}^\star$ to $\mathcal{G}^\star$ such that for all $s \in \mathcal{G}^\star$, $\rho(s) \in \mathcal{G}^\star$ is identical to s in all places except one, i.e. if $s = u_1 \cdots u_m$ and $\rho(s) = v_1 \cdots v_m$, then it is the case that for all but one $1 \leq i \leq m$, $u_i = v_i$.

We assume that the FEE framework is given an arbitrary but fixed set of edit operators. In our examples, we will assume a single family of edit operators — but we emphasize that this is just one example of the types of edit operators used in FEE. In our definitions, we assume that T_G is enumerated as $t_1, \ldots, t_k$ in some arbitrary but fixed order.

Definition 3 (Edit Operator ρ). The edit operator $\rho_{t_i,j,t_r}(s)$ replaces the j'th occurrences of t_i in s by t_r.

Though this may seem to be just one edit operator, it is an exceedingly powerful one as it is parametrized by 3 parameters i, j, r.

Example 4. Let us return to Example 1 and let s be the equation: $x_1 = 3 \times x_2 - 4 \times x_3 + 5 \times x_4 \times x_4$.
The result of applying the edit operator $\rho_{3,1,8}(s)$ is the equation $x_1 = 8 \times x_2 - 4 \times x_3 + 5 \times x_4 \times x_4$.
The result of applying the edit operator $\rho_{\times,2,+}(s)$ is $x_1 = 3 \times x_2 - 4 + x_3 + 5 \times x_4 \times x_4$.

We now illustrate the application of edit operators to differential equations.

Example 5. Let us return to Example 2 and let s be the equation $\partial(y_1 + x_1 \times x_1)/\partial x_1 = 3 \times x_2$.
The result of applying the edit operator $\rho_{\partial/\partial(x_1),1,\partial/\partial(x_3+2)}(s)$ is the equation $\partial(y_1 + x_1 \times x_1)/\partial(x_3 + 2) = 3 \times x_2$.
The result of applying the edit operator $\rho_{\times,2,\div}(s)$ is the equation $\partial(y_1 + x_1 \times x_1)/\partial x_1 = 3 \div x_2$.

The above example only shows two possible edit operators for a simple differential equation. Of course, many others are possible and FEE can work with *any* set of edit operators that users may design. However, when designing edit operators, one factor should be kept in mind: the ∂/∂ operator needs to specify the variable whose derivative is being taken. We now show the edit operators applied to transcendental equations.

Example 6. Let us return to Example 3 and let s be the equation $\sin x_1 = \exp(2 + x_3\hat{}(x_2 - 1))$.
The result of the edit operator $\rho_{\hat{},1,+}(s)$ is the equation $\sin x_1 = \exp(2 + x_3 + (x_2 - 1))$.
The result of edit operator $\rho_{\exp,1,\log_e}(s)$ is the equation $\sin x_1 = \log_e(2 + x_3\hat{}(x_2 - 1))$.

The above example only denotes two possible edit operators for a simple transcendental equation. While users can design their own edit operators, one thing to note is that: if t_i is an operator, then t_r should also be an operator taking the same number of variables — otherwise a unary operator may be replaced with a binary operator or vice-versa, resulting in equations that are not accepted by the CFG.

Definition 4 (Edit Sequence). An *edit sequence* is a finite sequence of edit operations. The result of applying the edit sequence $e_1, \ldots, e_m$ to an equation s is given by $e_m(e_{m-1}(\cdots e_1(s) \cdots))$.
Suppose $e_h = \rho_{t_i^h, j^h, t_r^h}$. The above edit sequence is said to be *singular* iff for all $1 \leq u, v \leq m$ such that $u \neq v$, it is the case that $(t_i^u \neq t_i^v \vee j^u \neq j^v)$.

Informally speaking, a singular edit sequence never contains two edits of the same terminal symbol. There is no loss of generality in restricting interest to singular edit sequences because if two edits e_u, e_v are such that $(t_i^u = t_i^v \wedge j^u = j^v)$, then the later of the two edits will generate the final result which means that the first of the two edits in the sequence can be deleted from the edit sequence without changing the final result of applying the edit sequence.

Definition 5 (Cost Function). A cost function cost is a mapping from edit operations to the set $\mathbb{R}^+$ of positive real numbers.
Cost functions can be applied to edit sequences in the obvious way by setting $\mathsf{cost}(e_1, \ldots, e_m) = \Sigma_{i=1}^m \mathsf{cost}(e_i)$.

The cost of an operation is intended to capture the visual difference between equations before and after the manipulation operation is applied. We would like an equation with the operation applied to be as similar as possible to what it was like before — otherwise attackers might easily discover that the equation (and hence the document it is in) is fake. Our FEE system seeks to minimize these costs, subject to generating fake equations sufficiently semantically different from the original.

Overall Goal (Informal). The overall goal of the FEE framework is the following: given an input equation E, an integer $k \geq 1$, a budget $B > 0$, and some "semantic constraints", find a set of k edit sequences $es_1, \ldots, es_k$ such that $\Sigma_{i=1}^{k} \mathsf{cost}(es_i)(E) \leq B$ (i.e. the budget constraint is satisfied) and such that a given objective function is optimized.

However, we have thus far not defined two important terms used in the above statement of the goal. First, every equation has some semantics — for instance, the equation $y = 2 \times x + 3$ has a semantics, namely it describes the straight line of slope 2 passing through the point $(0, 3)$. We cannot replace this equation with a "fake" equation that is syntactically within the stated cost bounds if the fake equation has a semantics that is dramatically different because an adversary would be able to easily detect the fact that the fake equation is a fake. However, a fake equation that uses the straight line $y = 1.9 \times x + 3.3$, on the other hand, might be close enough semantically to the equation $y = 2 \times x + 3$. In addition, we want our system of fake equations to be optimal in some sense. This sense could include deviating sufficiently from the original equation to be incorrect (so the adversary is faked out), but at the same time being sufficiently similar to the original equation to not obviously be a fake. Thus, the "quality" of a potential set of fake equations needs to be evaluated in some way via an objective function. These two points will be addressed in the next section which will formally define the problem of finding k fake equations as an optimization problem.

Example 7. Let us return to Example 4 with s being $x_1 = 3 \times x_2 - 4 \times x_3 + 5 \times x_4 \times x_4$. Without loss of generality, we assume that the cost of each edit operator is 1.
Then the cost of the edit sequence $es = \langle \rho_{3,1,8}(s), \rho_{\times,2,+}(s) \rangle$ is $\mathsf{cost}(es) = 2$ and the resulting fake equation is $es(s) = x_1 = 8 \times x_2 - 4 + x_3 + 5 \times x_4 \times x_4$.

Example 8. Let us return to Example 5 with s being $\partial(y_1 + x_1 \times x_1)/\partial x_1 = 3 \times x_2$. Assume that the cost of editing differential function is 2, otherwise 1.
Then the cost of the edit sequence $es = \langle \rho_{\partial/\partial(x_1),1,\partial/\partial(x_3+2)}(s), \rho_{\times,2,\div}(s) \rangle$ is 3 and the equation after edition is $\partial(y_1 + x_1 \times x_1)/\partial(x_3 + 2) = 3 \div x_2$.

Example 9. Let us return to Example 6 with s being $\sin x_1 = \exp(2 + x_3\hat{}(x_2 - 1))$. Assume that the cost of editing $t_i \in \mathbb{F}$ is 2, otherwise 1.
Then the cost of the edit sequence $es = \langle \rho\hat{}_{,1,+}(s), \rho_{\exp,1,\log_e}(s) \rangle$ is 4 and the new equation is $\sin x_1 = \log_e(2 + x_3 + (x_2 - 1))$.

The examples of three kinds of equations shown above demonstrate how edit sequences can change an equation with an associated cost. Costs are inputs (provided by the user) to the FEE framework.

5 FEE Algorithm

Given an equation E, integer $k > 0$ and budget B, we have designed an iterative algorithm to find fake equations one by one as shown in Algorithm 1.

FEE uses the two sets FE and ES respectively to store the generated fake equations and corresponding edit sequences (Line 1). FEE then proceeds iteratively till it has found the desired number (k) of fake equations. In each iteration, it formulates an optimization problem whose minimal cost solution returns an edit sequence $es \notin ES$ (Line 3). If this solution has a cost that fits within the overall budget, then it is added to the set of fake equations FE and the corresponding edit sequence is added to ES. Thus, we solve FEE[OPT] repeatedly until k fake equations are generated or the budget B is exhausted (Lines 2 - 8).

FEE's Run-time. The run time of Algorithm 1 is $O(kT_P)$, where k is the desired number of fake equations and T_P is the time required to solve FEE[OPT]. We will discuss T_P shortly.

In the rest of this section, we focus on the formulation of the optimization problem FEE[OPT] which lies at the very heart of the FEE algorithm.

Algorithm 1: FEE framework algorithm

```
Input: E, k, B
Output: FE
1 ES, FS ← ∅ // Initial set of edit sequences
   ES and fake equations FE ;
2 while |ES| < k do
3     es, cost(es) ← Solve FEE[OPT]// Get the
       solution es with minimum cost(es) by
       solving FEE[OPT] ;
4     if cost(es) + Σ_{es'∈ES} cost(es') ≤ B then
5         ES ← ES ∪ {es}, FE ← FE ∪ {es(E)}
6     else
7         Print "Budget not sufficient to generate k fake
           equations!"
8         break
```

5.1 FEE[OPT] Optimization Problem Formulation and the FEE-FAST Algorithm

Recall from the definition of an edit operator ρ_{t_i,j,t_r} on an equation E that t_i and t_r are both from the finite set T_G and E is treated as a fixed-length string. Therefore, for any equation E that we wish to generate fakes for, there exists a set $\Theta(E)$ of edit operators. When E is clear from context, we will write Θ instead of $\Theta(E)$.

We are interested in only singular edit sequences, i.e. for all $\rho_{t_i,j,t_r} \in \Theta$ with identical (t_i, j) values, *at most one* of them gets used in any edit sequence es. This assumption leads to no loss of generality. Suppose $\Theta = \{\Theta_{t_i,j}\}$, where each $\Theta_{t_i,j} = \{\rho_{t_i,j,t_r}\}$ is the set of edit operators for $(t_i, j) \in E$. Thus, an edit sequence es can be represented as a vector $es = \{k_{t_i,j}\}_{(t_i,j)\in E}$, where $k_{t_i,j}$ is an integer in the interval $[0, |\Theta_{t_i,j}|]$. When $k_{t_i,j} = 0$, es does not change (t_i, j). Otherwise, it uses the $k_{t_i,j}$th element of the set $\Theta_{t_i,j}$.

The goal of solving the optimization problem FEE[OPT] is to find the best edit sequence $es \notin ES$ while minimizing $\mathsf{cost}(es)$.

Before formally writing down the optimization problem FEE[OPT], we note that we wish to generate fake equations that will deceive the adversary. However, thus far, the notion of edit operators and edit sequences do not consider the semantics of the equations. If we ignore the semantics of the equation E for which we are generating k fake equations $fe_1, \ldots, fe_k$, it may become very easy for an adversary to infer that $fe_1, \ldots, fe_k$ are fake. For instance, suppose we wish to generate just one fake version of the equation $y = 2 \times x + 3$. Replacing this with the equation $y = 2\hat{}x + 3$ may be obtainable with a valid edit sequence, but because this equation's semantics is dramatically different from that of the original, this may be easily detectable to an adversary.

Thus, the optimization problem needs to describe a set of constraints that would achieve a user-specified desired level of deception of the adversary and we describe some of these constraints below.

1) *Constraints pertaining to universal truths, C_{tru}:* These are constraints which examine whether any variable in the equation contains any well-known phenomenon and pertain to information from "universal truths" in common knowledge. For example, if $\log 2 \times x_1$ with x_1 representing the mass of an object is in the equation E, then there is a corresponding universal truth constraint which should eliminate ess that would generate fake equations with components like $\log -2 \times x_1$. Universal truth constraints are also used to ensure that manipulated equations continue to maintain the unit consistency in both sides of the equation. For example, if the left hand side of the equation is a mass value in kilograms, the right side should also maintain the same unit. We denote the set of ess that violate the universal truth as $ES_{t\bar{r}u}$.
2) *Constraints pertaining to the form of an equation C_{nat}:* An original equation can be one of many types (e.g., polynomial, differential or transcendental). This constraint specifies whether the nature of the original equation should be maintained. These constraints can be taken into consideration while generating the set of edit operators Θ.
3) *Constraints pertaining to metrics in model hypothesis, C_{hyp}:* In the case where the original equation is a model fitted on some given/inferrable dataset, the hypothesis $\varphi(E)$ (e.g., $t-$test) might be discussed. The user may require that the new equation $es(E)$ also satisfies the hypothesis, i.e., $\varphi(es(E)) \geq \iota_\varphi$, where ι_φ is a corresponding threshold (e.g., 5% significance $t-$test value).
4) *Constraints pertaining to model metrics, C_{met}:* In the case that the original equation is a model fitted on some given/inferrable dataset, the model metric $\phi(E)$ (e.g., the coefficient of determination R^2) should not vary dramatically between the original and the fake(s), i.e., $|\phi(E) - \phi(es(E))| \leq \iota_\phi$, for some $\iota_\phi \geq 0$. When $\iota_\phi = 0$, the fake model would be required to have the same value of the associated metric (e.g., coefficient of determination) as the original model. In practice, the value of the threshold ι_ϕ that the metric can deviate by will be set as a constraint by the system security officer who manages the FEE framework.
5) *Constraints pertaining to enough distance on model prediction, C_{prd}:* When the original equation is a model fitted on some given/inferrable dataset, we would like to protect the model prediction on the whole variable space so that an adversary who uses the fake equation receives erroneous results. We can therefore define a distance function $D_{prd}(E, es(E))$ between the predictions of the two equations $E, es(E)$ and require the distance to fall within a user-specified interval $[L, U]$ in order to ensure that the manipulated equation is distinct from the original one, but still within a suitable user-specified range.
6) *Constraints pertaining to consistency with other occurrences of equation terms in the context, C_{css}:* While we assume that only one equation is manipulated at a time in this paper, a real world document may contain multiple equations that depend upon each other. We must therefore examine other occurrences of the *term*s $\hat{s}$ in the equation E and try to keep the number of times there is an inconsistency with other equations within a given user-specified range that makes it hard for the adversary to identify the equation $es(E)$ as fake. It is also possible to set this number to 0. We define $N_{vio}(\hat{s})$, the violation frequency of a *term* to the number of other occurrences of this *term* in the context if it is influenced by the edit sequence, and set a max number N_{vio}^{max} for the equation $es(E)$.

We are now finally in a position to present the formulation of the optimization problem FEE[OPT].

$$\begin{aligned}
\textsf{FEE[OPT]} \quad \min_{es \notin ES} \quad & \textsf{cost}(es) && (1)\\
\text{s.t.} \quad & es \notin ES_{t\bar{r}u} && (2)\\
& \varphi(es(E)) \geq \iota_\varphi && (3)\\
& |\phi(E) - \phi(es(E))| \leq \iota_\phi && (4)\\
& D_{prd}(E, es(E)) \in [L, U] && (5)\\
& \sum_{\hat{s} \sqsubseteq E : \exists k_{t_i,j} > 0, \forall (t_i,j) \in \hat{s}} N_{vio}(\hat{s}) \leq N_{vio}^{max} && (6)
\end{aligned}$$

Note that $ES_{t\bar{r}u}$ denotes the set of ess that violate the universal truths of the equation E. The $\exists k_{t_i,j} > 0, \forall (t_i, j) \in \hat{s}$ in the last constraint means that the *term* has been modified by some edit operator of es.

In order to solve FEE[OPT], let us analyze the problem step by step. First, this problem has a set of integer variables, i.e., the $k_{t_i,j}$s in es indicating whether a given edit operator (t_i, j) is modified and if yes, which t_r is used to replace it. Suppose we define a constant cost for each edit operator in Θ. Then the objective function would be linear w.r.t. the costs of es. For *constraints pertaining to universal truths* (Equation (2)), it is reasonable to assume that there are a finite number of natural truths for equation E. We therefore need to derive the set $ES_{t\bar{r}u}$, which means that we must enumerate and check all the possible edit sequences $es \notin ES$. If the equation E consists of N_t editable pairs (t_i, j) ($N_t \leq |\Theta|$), the complexity of deriving $ES_{t\bar{r}u}$ is at least $O(2^{N_t})$ and at most $O(2^{|\Theta|})$. When the equation E is a regression model, in

order to handle constraints (3), (4) and (5) , we need to first **transform equation $es(E)$ from the string representation to a mathematical representation**, such that the required calculations (w.r.t. hypothesis, metrics and predictions) can be performed. However, to the best of our knowledge, there is no direct transformation for equations from a string expression to a mathematical expression. Instead, what we can do is to design a function to realize it. This limitation makes it impossible to directly solve the problem using existing solvers and algorithms when constraints (3) —(5) exist. Finally, for constraint (6), we first enumerate the set of *terms* of equation E, get the number of other occurrences for each *term* with the document context, then compute the number of violations for each edit sequence and compare it with the threshold.

Algorithm 2: FEE-FAST Approximate algorithm for solving FEE[OPT]

Input: E, ES, Θ
Output: es

1 $ES' \leftarrow \emptyset$ // Initialize the set of candidate edit sequences
2 **for** *each* $es \notin ES$ **do**
3 **if** $es(E) \notin ES_{t\bar{r}u}$ **then**
4 $ES' \leftarrow ES' \cup \{es\}$
5 $ES' \leftarrow \{es'_1, \ldots, es'_{|ES'|}\}$ such that $\mathsf{cost}(es'_1) \leq \cdots \leq \mathsf{cost}(es'_{|ES'|})$ // Sort elements of ES' in **ascending order** according to their costs
6 $es = [\,]$ // Initialize the solution as an empty vector
7 **for** $l = 1 : |ES'|$ **do** // Loop starts from the first element
8 Get the mathematical formulation of $es'_l(E)$
9 **if** $es'_l(E)$ *satisfies constraints* (3), (4), (5) *and* (6) **then**
10 $es \leftarrow es'_l$
11 **break**
12 **if** $es = [\,]$ **then**
13 Print "No eligible edit sequence exists!"

We now propose Algorithm 2 to solve the problem **P**. The current set ES of selected edit sequences and the set Θ of edit operators are inputs for the generation of the output es, i.e., the best eligible (unused) edit sequence with the lowest cost. First, the set ES' of candidate edit sequences is initialized to the empty set (Line 1). We then add each edit sequence that obeys the universal truths to ES' (Lines 2 - 4)[3]. The elements in ES' are then sorted in ascending order according to their costs (Line 5). We then initialize the output es as an empty vector (Line 6) and check the candidate edit sequences in ES' one by one (Lines 7 - 11) until a satisfying one is found (Lines 9 - 11) or all candidates are found to be ineligible (Lines 12 - 13). Note that the cost function $\mathsf{cost}(\cdot)$ is called only in Line 5 — hence the correctness of our algorithm is not affected by the complexity (or non-linearity) of the cost function. However the run-time could be affected because $\mathsf{cost}(\cdot)$ is not limited at all in what it can be and hence a subroutine implementing it could, in theory, be expensive.

Run-Time of FEE-FAST. The run time of Algorithm 2 is $O(|ES'|)$, which means it depends on the number of remaining eligible edit sequences. Users may use additional steps to further constrain $|ES'|$ in practice.

While the formulation of FEE[OPT] can vary a great deal for different applications, the proposed framework can always be adapted according to user specifications. In the next section, we demonstrate how to apply FEE framework under 3 distinct realistic circumstances and conduct a detailed human evaluation on 20 different documents.

3. In case the number of valid edit sequences is enormous, enumerating the set ES' is unrealistic. We then instead use genetic approaches to generate edit sequences starting from the ones with lower costs.

6 Experiments

We collected a set of 20 patents ranging over diverse areas that contained different kinds of equations. We then used the FEE framework to generate 10 fake versions for one equation from each document. We then invited a panel of 50 human subjects to identify the original correct equation. Thus, each human subject was given 20 tasks with each task containing 11 versions (10 fake, 1 real) of an equation.

To apply FEE on a document with a specific equation, the process includes following steps:

- Step 1 Decide the content of set T_G and the initial set of edit operators Θ;
- Step 2 Check the constraints presented in Section 5.1, reduce the set Θ and get constraints in the FEE[OPT] optimization problem;
- Step 3 Customize the hyper-parameters if needed, otherwise use the default values;
- Step 4 Run Algorithms 1 and 2 to generate the set FE of fake equations.

While Steps 1 and 2 require human inputs, Steps 3 and 4 are fully automated. Note that the parameter setting in Step 3 is flexible as FEE system users can also define their own parameters. Sections 6.1 and 6.2 demonstrate the process and result of applying FEE each of the above steps for 2 representative documents with 2 different equations.

6.1 Linear Equation Manipulation

[8] presents a multiple linear regression model to estimate the productivity of construction (i.e. building) operations that use concrete. The linear regression model is fitted on a set of data collected from a major civil engineering project. The authors present some statistics for the data, as well as hypotheses and metrics associated with their final model. We select the linear equation E they use in their paper

$$P_{actual} = \quad 1.31T_p + 1.75V_a + 0.56T_n + 0.59W - 0.01C_t + 0.37L_n - 6.95$$

and try to generate fake versions of it.

Step 1: To start using FEE, we must first decide $T_G = \mathbb{R} \cup \mathbb{O} \cup \mathbb{X}$ of the grammar for this equation. Let $\vec{c} = \{c_1, ..., c_7\}$ denote the coefficient vector in equation E. Considering that coefficients in $\vec{c}$ are all 2-decimal positive numbers, we set $\mathbb{R}$ as the set of all 2-decimal non-negative numbers with upper bound $\max_i c_i + \sigma_{\vec{c}} = 9.36$ and lower bound $\max(0, \min_i c_i - \sigma_{\vec{c}}) = 0.00$, where $\sigma_{\vec{c}}$

is the standard deviation of $\vec{c}$. The reason for setting these bounds is to limit how the coefficients are manipulated so that the fake equations will not have unreasonably complex coefficients. The operators are set to $\{+, -, \times\}$, $\mathbb{O}$ as these are the ones that appear in the equation. Furthermore, $\mathbb{X} = \{P_{actual}, T_p, V_a, T_n, W, C_t, L_n\}$. To ensure $\rho_{t_i,j,t_r}(E)$ is still an equation accepted by the grammar, the initial set Θ of edit operators is

$$\Theta = \{\ \rho_{t_i,j,t_r} : t_i \neq t_r\ \&\ (\ t_i, t_r \in \mathbb{O}\ ||\ t_i, t_r \in \mathbb{R} \cup \mathbb{X}\ )\ \}$$

Step 2: Check each constraint presented in Section 5.1. The discussion of each constraint is listed below.

1) Although all the elements in $\mathbb{X}$ are supposed to be positive values, there is no need for universal truth constraints because the edit operators will not lead to any results with violations to these universal truths.
2) For this example, suppose we would like to maintain a *linear model* as this is mentioned in the context of the document. In this case, we only modify the coefficients and the operators $+, -$. When a coefficient is changed to 0, it means the corresponding variable is not considered in the linear model. Thus, Θ is updated as follows:

$$\Theta = \{\ \rho_{t_i,j,t_r} : t_i \neq t_r\ \&\ (t_i, t_r \in \{+, -\}\ ||\ t_i, t_r \in \mathbb{R})\}$$

Note that the size of Θ is $937^7 \times 2^7$.
3) $t-$test and $F-$test results are discussed in the paper. To add corresponding constraints for these hypothesis, we first generate synthetic data used for model fitting with the statistics provided in the document (details in Appendix A), then set corresponding constraints for them.
4) As in the case of constraint 3, we set constraints for model metrics R^2 with synthetic data.
5) We derive a reasonable interval for each independent variable in $\mathbb{X} - \{P_{actual}\}$, then sample $1M$ points in the united $6-$dimensional space and evaluate the difference of model predictions on them.
6) The set of *term*s of the original equation E that occur in the context of the document is empty.

Step 3: We set the cost of $\rho_{t_i,j,t_r} = |t_i - t_r|$ while $t_i, t_r \in \mathbb{R}$; $\max_{t_1,t_2 \in \mathbb{R}} |t_1 - t_2|$ if $t_i, t_r \in \mathbb{O}$. The intuition is that (1) a larger difference in the value of a coefficient increases the amount of visual difference and (2) changing the operator basically changes the relationship between the independent variable and P_{actual}, thus it can be more significant than changing the value of a coefficient. Our goal is to generate $k = 10$ fake equations with budget $B = 60$. The threshold ι_φ is $t(0.025; 192)$ and $F(0.01, 8, 192)$ respectively for $t-$test and $F-$test; $\iota_\phi = 0.03$ for R^2. We try different values for L and U. Distance function D_{prd} takes the sum of absolute distance of predictions on generated data points.

Step 4: We used R to develop the algorithm. Considering that the variable space is enormous, instead of generate and rank the set ES' (Lines 1 to 5 of Algorithm 2), we directly start searching from edit sequences with smaller costs.

6.1.1 Generated Fakes

We generated 10 fake equations as shown in Table 1 — we only show the right hand side of the equations as all the left hand sides are the same. The R^2 of all generated equations is in $[0.769, 0.841]$ and the cost varies from 0.5 to 8.21.

6.2 Differential Equation Manipulation

[7] models gene expression using differential equations. This paper includes analytical discussion rather than statistical analysis for its equations. We take one differential equation as the target E for which we wish to generate fakes:

$$\frac{d^2\mathbf{r}}{dt^2} = (-CUC^{-1} - V)\frac{d\mathbf{r}}{dt} + (-CUC^{-1}V + CL)\mathbf{r}$$

Step 1: We extract all the notations related to the equation E to get the set $T_G = \mathbb{R} \cup \mathbb{O} \cup \mathbb{X}$. The current coefficients in E are all 1 and we get $\mathbb{R} = \{1, 2, 3\}$; $\mathbb{O} = \{+, -, \times, d/d, {}^{-1}, {}^{T}\}$, where ${}^{-1}$, ^{T} are inverse and transposition operators for matrices; $\mathbb{X} = \{t, \vec{r}, \vec{p}, \vec{s}, \vec{x}, \vec{\lambda}, L, V, U, C, M\}$. To ensure that fake equations are accepted by the grammar and still remain syntactically valid differential equations, we initialize

$$\begin{aligned}\Theta = \{\ \rho_{t_i,j,t_r} :\quad & t_i \neq t_r\ \& \\ (\quad & t_i, t_r \in \mathbb{R} \cup \mathbb{X}\ || \\ & t_i, t_r \in \mathbb{O}^1 = \{\ {}^{-1},\ {}^{T}\}\ || \\ & t_i, t_r \in \mathbb{O}^2 = \{d/d, +, -, \times\}\ )\ \}\end{aligned}$$

Step 2: Constraints 3 to 5 are not applicable. Other constraints are:

1. The equation E involves a matrix, so we need to keep the dimension of the matrices consistent. From the document we get: vectors $\vec{r}, \vec{p}, \vec{s}$ are $n-$dimensional, $\vec{x}, \vec{\lambda}$ are $2n-$dimensional, matrices L, V, U, C are $n \times n-$dimensional and M is $2n \times 2n-$dimensional. Thus FEE should ensure that the dimension of both sides of E is the same. To achieve this, we build a function to check this for $es(E)$.
2. To ensure that the generated fake equation is still a differential equation, we must ensure that *at least one differential operator remains in place after the edit*. This can be achieved by using the constraint $\exists k_{d/d,j} = 0$. There is no need to update Θ.
6. The set of *term*s with non-zero number of other occurrences are: $\{CUC^{-1}, -CUC^{-1} - V, CL, C^{-1}V, (-CUC^{-1}V + CL)\vec{r}, \frac{d^2\mathbf{r}}{dt^2}, (-CUC^{-1} - V)\frac{d\mathbf{r}}{dt} + (-CUC^{-1}V + CL)\mathbf{r}\}$.

Step 3: We set the cost for all edit operators to the same value 1 and generate $k = 10$ fake equations with budget $B = 50$. The $N_{vio}(\hat{s})$ value for each *term* is $2, 1, 2, 3, 1, 1, 1$ respectively. N_{vio}^{max} is set as 10.

Step 4: We use Python to develop the algorithms and find eligible fake equations.

6.2.1 Generated fakes

The FEE framework generated the 10 fake equations shown in Table 2. The cost varies from 3 to 6. The corresponding edit sequences are also listed in the table.

TABLE 1
Right hand sides of fake equations with model metrics for Section 6.1. "P_{actual} =" is the left hand side of all the equations and hence is not shown explicity in the table.

Fake Equation	R^2	F-stat	Distance	Cost	Cost to R^2
2.3924T_P +0.0511V_a -0.0796T_n +0.0394C_t -0.2651W -0.2008L_n +5.0218	0.840	170.525	114.253	0.50	1.68
1.7569T_P+0.0561V_a-0.1140T_n -0.0193C_t-0.2434W+ 0.3196L_n+ 8.8719	0.833	162.280	107.551	1.53	0.54
2.3150T_P +0.0505V_a -0.0723T_n +0.0300C_t -0.0079W -0.1714L_n +4.9161	0.841	172.240	110.632	4.96	0.17
2.3250T_P +0.0525V_a -0.0713T_n +0.0320C_t -0.0081W -0.1624L_n +4.9161	0.841	172.240	113.632	5.04	0.17
1.7456T_P+0.0577V_a-0.164T_n -0.0221C_t-0.2554W+ 0.3196L_n+ 8.8719	0.833	164.270	109.591	5.20	0.16
1.8480T_P+ 0.0563V_a -0.0982T_n-0.0115C_t -0.0467W +0.1550L_n+7.8201	0.835	163.130	105.664	5.36	0.15
3.6712T_P +0.0181V_a+0.1642T_n+ 0.1373C_t +0.0050W -1.4549L_n -1.6103	0.769	106.830	124.24	6.18	0.12
2.5871T_P +0.0382V_a+0.0352T_n +0.0516C_t+0.0173W -0.5264L_n +3.7194	0.837	166.922	101.545	7.45	0.11
1.5208T_P +0.0594V_a -0.1411T_n -0.0339C_t -0.6544W+0.5576L_n+ 11.0711	0.792	120.601	153.775	7.74	0.10
2.3224T_P +0.0512V_a -0.0768T_n +0.0344C_t -0.2851W -0.2008L_n +5.0218	0.840	169.525	109.253	8.21	0.11

TABLE 2
Fake differential equations with their corresponding costs and edit sequences for Section 6.2.

Fake Equation	Edit Sequence	Cost
$\frac{d^2r}{dt^2} = (-CUC^{-1} - V^{-1})\frac{dr}{dt} + (-CUC^{-1} + V + CL) - r$	$\{\rho_{V,1,V^{-1}}; \rho_{\times,6,+}; \rho_{\times,8,-}\}$	3
$\frac{d^2r}{dt^2} = (-CUL - V)\frac{dr}{dt} + (CUC^{-1} + V + CL)r$	$\{\rho_{C^{-1},1,C}; \rho_{-,3,+}; \rho_{\times,6,+}\}$	4
$\frac{d^2r}{dt^2} = (-CUC^{-1} - V)\frac{dr}{dt} + (-CU^TV^{-1}V + UL)r$	$\{\rho_{U,2,U^T}; \rho_{C^{-1},2,V^{-1}}; \rho_{C,3,U}\}$	4
$-\frac{d^2r}{dt^2} = (-C + UC^{-1} - V)\frac{dr}{dt} + (-CUC^{-1} - V + CL) - r$	$\{\rho_{+,1,-}; \rho_{\times,1,+}; \rho_{\times,6,-}; \rho_{\times,8,-}\}$	4
$\frac{d^2r}{dt^2} = (-CUC^{-1} - V^{-1})\frac{d^2p}{dt^2} + (-CUC^{-1}V - CL)r$	$\{\rho_{V,1,V^{-1}}; \rho_{+,3,-}; \rho_{\frac{dr}{dt},1,\frac{d^2p}{dt^2}}\}$	5
$\frac{d^2r}{dt^2} = (-CUC^{-1} - V)\frac{dr}{dt} + (-C^{-1}U^TC^{-1} + V + C + L) - r$	$\{\rho_{C,2,C^{-1}}; \rho_{U,2,U^T}; \rho_{\times,6,+}; \rho_{\times,7,+}; \rho_{\times,8,-}\}$	5
$\frac{d^2r}{dt^2} = (-C - UC^{-1} - L^{-1})\frac{dr}{dt} + (-L^{-1}UC^{-1}V + CL)r$	$\{\rho_{V,1,L^{-1}}; \rho_{C,2,L^{-1}}; \rho_{\times,1,-}\}$	5
$\frac{d^2r}{dt^2} = (-CUU - V)\frac{dr}{dt} + (-CUC^TV + CU)r$	$\{\rho_{C^{-1},1,U}; \rho_{L,1,U}; \rho_{C^{-1},2,C^T}\}$	5
$\frac{d^2r}{dt^2} = (-CUC - V)\frac{d^2s}{dt^2} + (C + UC^{-1}V + CL)r$	$\{\rho_{C^{-1},1,C}; \rho_{\frac{dr}{dt},1,\frac{d^2s}{dt^2}}; \rho_{\times,4,+}; \rho_{-,3,+}\}$	6
$\frac{d^2r}{dt^2} = (-V^{-1}UC^{-1} - C)\frac{dr}{dt} + (-CU + C^{-1}V + CV^{-1})r$	$\{\rho_{C,1,V^{-1}}; \rho_{V,1,C}; \rho_{L,1,V^{-1}}; \rho_{\times,5,+}\}$	6

6.3 Human Evaluation

As mentioned earlier, we selected 20 technical documents on diverse topics. We selected one equation from each document as the target equation for manipulation. The equation length (which is denoted by the number of editable components in the equation) varies from 8 to 34. We used FEE to generate 10 fake equations per equation which were then used to generate 10 fake documents. Thus, each of the fake documents had exactly one fake equation in it — and the subjects were told that. This gives them a potential advantage in detecting fakes. For each original/fake document, we show each subject 3 pages in all: the page on which the equation occurs together with the immediately preceding and immediately succeeding page. This gives the human subject some context. *Note that showing 3 pages surrounding an equation biases the experiment in favor of the adversary by providing him very valuable context in the form of text* that the FEE algorithm does not currently consider to modify.[4]

Our experiment involves 50 workers on Amazon Mechanical Turk (MTurk), all of whom are required to have a Master's degree or higher from the United States. Each worker is asked to answer 20 questions in a randomized order. Each question has 11 randomized choices and each choice is a 3-page document. Exactly 1 of the 11 choices is the original document. All workers were told that only one equation in a fake document is fake (this is a tougher test for our FEE system because the subjects know that they only need to find one fake equation) and they are asked to select the document they think is most likely to be fake, 2nd most likely to be fake, and 3rd most likely. We name these choices the 1st, 2nd and 3rd choice respectively.

Deception Rate of $Hit@1$. We define the *Deception Rate* as the probability that the original document is not discovered as the 1st choice when it is mixed with 10 generated fake

4. Combining FEE with a very robust paradigm for generating fake textual content such as the FORGE system [5] is an important next step that we propose to study in future work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3038132, IEEE Transactions on Dependable and Secure Computing

versions. On average, this deception rate is 88.6%.

Distribution of $Hit@1$. Figure 2 shows the distribution of $Hit@1$. For each human subject, we first calculate $Hit@1$ as the number of times that the original document was selected as their 1st choice. $Hit@1$ is a number between 0 through 20. We then plot Figure 2 with the above data. The x-axis shows the number of documents (0 through 20) and the y-axis shows the portion of workers whose 1st choice correctly identified that amount of original documents. It is shown that $Hit@1$ varies from 0 to 10. We fit the distribution with a normal distribution and draw the probability distribution function (PDF) curve in red. The average value of $Hit@1$ is 2.28 with a standard deviation 2.03. This means that on average, each worker was only able to correctly get 2.28 real documents in 20 guesses.

Deception Rate of $Hit@1 + Hit@2$. We also looked at what happened when we considered a worker's guess to be correct if either his 1st or 2nd choice was correct. On average and as expected (as we are more generous in accepting the worker's guess as correct in this case), the deception rate decreases to 80.7% from 88.6% in the case of $Hit@1$.

Distribution of $Hit@1 + Hit@2$. Figure 3 shows the distribution of guesses by the 50 MTurk workers in the case when we consider either their 1st or 2nd guess to be correct. In this case, the number of correct selections varies from 0 to 12. The mean and standard deviation of $Hit@1 + Hit@2$ are respectively 3.86 and 2.55.

Deception Rate of $Hit@1 + Hit@2 + Hit@3$. We also looked at what happened when we considered a worker's guess to be correct if one of his top-3 choices turned out to be correct. The deception rate now falls to 70.9%.

Distribution of $Hit@1 + Hit@2 + Hit@3$. Considering all the top-3 choices (i.e., $Hit@1+Hit@2+Hit@3$), as shown in Figure 4, the mean number of documents uncovered under this very generous (to the adversary) setting is 5.82 with a standard deviation equal to 2.85.

From the analysis in the above 3 cases, we can conclude that FEE is able to deceive most adversaries. Furthermore, FEE will be able to do an even better job in deceiving the adversary if some of the following steps are followed.

1) We use publicly accessible patents as the original documents in the human evaluation and there is nothing that stops MTurk workers who have full access to the Internet from searching for the correct answer on the Internet. In practice, our proposed approach would most probably be used to protect private documents where adversaries cannot carry out such a search.
2) We only alter one equation for each document in the experiments. In practice, an operational system would generate fake equations and text simultaneously, making it much harder for an adversary to find inconsistencies between the two.
3) In the experiments, we truncate 3-pages from patents with dozens of pages and clearly tell the workers that only one equation is different among different choices. However, attackers in real-world scenarios will need to check the complete text of all documents, real and fake, and they will not know how many equations (or how much of the text) has been faked.

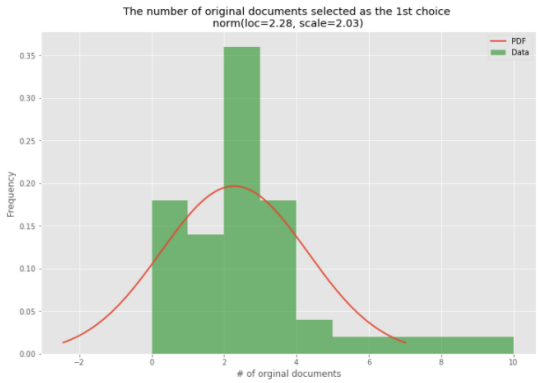


Fig. 2. The number of original documents selected as the 1st choice

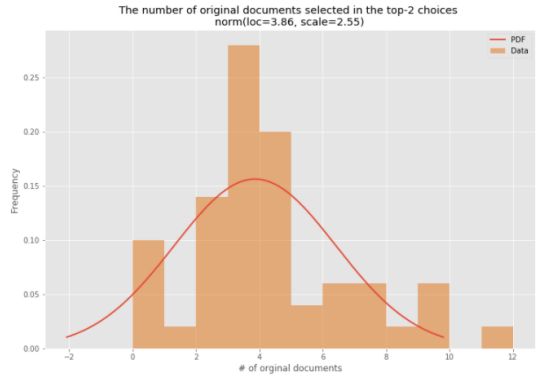


Fig. 3. The number of original documents selected by the first 2 choices

4) Finally, nothing prevents us from generating more than 10 fakes per original document. As the number of fake documents goes up, the probability that the adversary will be able to find the real one goes down.

Run-Time. All the fake equations in our experiments were generated in under 5 seconds. As a consequence, we did not run further run-time experiments as the fake equation generation process is clearly fast enough for practical use.

Limitations. Though we selected technical documents from several different fields to generate fake equations, this breadth of equations comes at a price. We were not able to select experts in the specific areas of those equations to

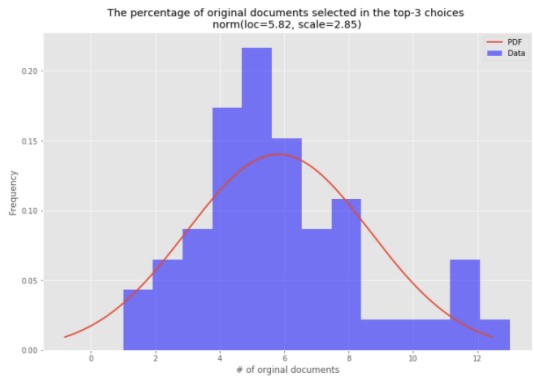


Fig. 4. The number of original documents selected by the first 3 choices

evaluate the quality of the fakes because selecting workers with specialized areas of expertise (e.g., degrees in genetics) on Amazon Mechanical Turk is not supported and is very challenging. It is therefore possible that the deception rates in this paper will go down a bit if true experts in the discipline of an equation are used for evaluation.

An Important Note. An alert reader might wonder how a legitimate user would distinguish a real document from one containing a fake equation. This problem has been solved in the FORGE system [5] which is why we do not discuss it in detail here. In a nutshell, it is possible to embed a message authenticating code in every document, both real and fake. An authorized user with a private key will be able to use his private key with the code in a document to determine whether the document is real or fake.

7 Usage of FEE and Next Steps

We conclude by noting the big picture underlying FEE.

A technical document d may contain diverse forms of content including text, tables, equations, formulas, flowcharts, diagrams, and more. Generating a fake version of d involves not only generating fake versions of each of these types of content, but also ensuring that they are combined together well. In past work [5], we have developed methods to generate fake versions of the textual part of d as well as the tables in d [6]. This effort shows how to generate fake versions of equations. In work currently undergoing a second round of review, we have proposed a unifying framework to integrate these different types of fakes using probabilistic logic graph [11].

Future steps revolve around generating fake versions of flowcharts and diagrams so that a comprehensive method to generate fake documents exists.

8 Conclusion

There has been considerable recent interest in addressing the problem of intellectual property theft by automatically generating multiple fake copies of every real document that an organization wishes to protect from IP thieves. This strategy imposes a cost on the adversary as s/he now needs to figure out which of many exfiltrated documents is real and which are fake.

Past work for the auto-generation of such fakes has focused on the textual part of a document. However, technical documents have many constituent parts including text, tables, equations, diagrams, and more. In this paper, we develop the methods needed to automatically generate multiple fake equations. Moreover, our FEE framework ensures that the semantics of the original equation is close enough to the original to be believable, yet sufficiently far enough to be likely to be wrong.

FEE uses a mechanism that iteratively solves an optimization problem in each iteration. However, the optimization problems solved by FEE are not traditional optimization problems (e.g. integer linear programs, knapsack problems, etc.). Rather, they involve a mix of numeric and logical constraints. We present a specialized algorithm, FEE-FAST, that solves these non-traditional optimization problems within each loop of FEE.

We have tested FEE out with a panel of 50 human subjects on 20 real world patents and shown that FEE has a high rate of deception, even when the subjects are provided some advantages when compared to FEE.

Limitation: we need to manually pre-process the constraints before feeding it to the FEE framework.

9 Acknowledgement

This work was supported in part by the Office of Naval Research grants N00014-18-1-2670, N00014-16-1-2896, and N00014-20-1-2407; and by the Army Research Office under grant W911NF-13-1-0421.

References

[1] M. M. R. Alavi Milani, S. Hosseinpour, and H. Pehlivan. Rule-based production of mathematical expressions. *Mathematics*, 6(11):254, 2018.

[2] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters*, 35:58–67, 2014.

[3] D. M. Bates and D. G. Watts. *Nonlinear regression analysis and its applications*, volume 2. Wiley New York, 1988.

[4] L. Bilge and T. Dumitraş. 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*, pages 833–844, 2012.

[5] T. Chakraborty, S. Jajodia, J. Katz, A. Picariello, G. Sperli, and V. S. Subrahmanian. FORGE: A Fake Online Repository Generation Engine for Cyber Deception. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[6] H. Chen, S. Jajodia, J. Liu, N. Park, V. Sokolov, and V. Subrahmanian. Faketables: using gans to generate functional dependency preserving tables with bounded real data. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2074–2080. AAAI Press, 2019.

[7] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. In *Biocomputing'99*, pages 29–40. World Scientific, 1999.

[8] P. Dunlop and S. Smith. Estimating key characteristics of the concrete delivery and placement process using linear regression analysis. *Civil Engineering and Environmental Systems*, 20(4):273–290, Dec. 2003.

[9] C. A. Fowler and R. F. Nesbit. Tactical deception in air-land warfare. *Journal of Electronic Defense*, 18(6):37–45, 1995.

[10] U. Garain and B. B. Chaudhuri. Recognition of online handwritten mathematical expressions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(6):2366–2376, 2004.

[11] Q. Han, C. Molinaro, A. Picariello, G. Sperli, V. Subrahmanian, and Y. Xiong. Generating fake documents using probabilistic logic graphs. *under review*, 2020.

[12] J. E. Hopcroft and J. D. Ullman. Formal languages and their relation to automata. 1969.

[13] S. Jajodia, N. Park, F. Pierazzi, A. Pugliese, E. Serra, G. I. Simari, and V. Subrahmanian. A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security*, 12(11):2532–2544, 2017.

[14] P. Karuna, H. Purohit, R. Ganesan, and S. Jajodia. Generating Hard to Comprehend Fake Documents for Defensive Cyber Deception. *IEEE Intelligent Systems*, 33(5):16–25, 2018.

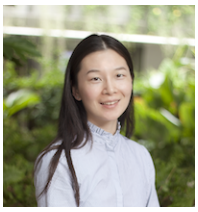
[15] D. Kushner. Digital decoys [fake mp3 song files to deter music pirating]. *IEEE Spectrum*, 40(5):27, 2003.

[16] C. L. Martin. Military deception reconsidered. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2008.

[17] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi. Preserving both privacy and utility in network trace anonymization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 459–474, 2018.

[18] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim. Data Synthesis based on Generative Adversarial Networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, June 2018. arXiv: 1806.03384.

[19] Y. Park and S. J. Stolfo. Software decoys for insider threat. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 93–94, 2012.
[20] A. Shabtai, Y. Elovici, and L. Rokach. *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media, 2012.
[21] B. Whitham. Automating the generation of fake documents to detect network intruders. *International Journal of Cyber-Security and Digital Forensics*, 2(1):103, 2013.
[22] B. Whitham. Automating the generation of enticing text content for high-interaction honeyfiles. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.
[23] B. Whitham, T. Turner, and L. Brown. Automated processes for evaluating the realism of high-interaction honeyfiles. In *Proceedings of the 14th European Conference on Cyber Warfare and Security*, page 307, 2015.
[24] C. Yang, Z. Wang, X. Zhu, C. Huang, J. Shi, and D. Lin. Pose guided human video generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 201–216, 2018.
[25] J. Yuill, M. Zappe, D. Denning, and F. Feer. Honeyfiles: deceptive files for intrusion detection. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 116–122. IEEE, 2004.



Yanhai Xiong is a Postdoc working in Dartmouth College since July, 2018. She received her PhD degree in Computer Science and Engineering from Nanyang Technological University, Singapore and the Bachelor degree in Automation from University of Science and Technological University of China. Her research interests lie in optimization, machine learning, cybersecurity and smart cities.

Giridhar Kaushik Ramachandran is currently a doctoral researcher in Information Sciences and Technology at George Mason University. He received his Bachelor of Science Honors degree in Mathematics in 2012 and Master in Business Administration degree in 2014 from the Sri Sathya Sai Institute of Higher Learning, India. He has over 4 years of diverse experience as a data scientist in the fields of finance, human resource management and education. He is interested in the applications of machine learning, optimization and statistics in cybersecurity, health and social media.

Rajesh Ganesan received the M.S. degree in industrial engineering, the M.A. degree in mathematics, and the Ph.D. degree in industrial engineering from the University of South Florida, Tampa, FL, USA, in 2002, 2005, and 2005, respectively. He is currently an Associate Professor of systems engineering and operations research with George Mason University, Fairfax, VA, USA, where he is with the Center for Secure Information Systems and the Center for Air Transportation Systems Research. His research interests include stochastic optimization (approximate dynamic programming), Bigdata analytics, multiscale statistical data analysis using wavelets, and engineering education. His research applications include cybersecurity, healthcare, defense, air transportation, and nanomanufacturing. Dr. Ganesan is a Senior Member of the Institution of Industrial Engineers and a member of the American Society for Engineering Education and INFORMS professional organization.

Sushil Jajodia is University Professor, BDM International Professor, and the founding director of Center for Secure Information Systems in the Volgenau School of Engineering at the George Mason University, Fairfax, Virginia. He is also the director of the NSF I/UCRC Center for Cybersecurity Analytics and Automation (now in Phase II). Before coming to Mason, he held permanent positions at the National Science Foundation; Naval Research Laboratory, Washington; and University of Missouri, Columbia. He has also been a visiting professor at the University of Milan, Sapienza University of Rome, Cambridge University, King's College London, Paris Dauphine University, and Imperial College. Dr. Jajodia received his PhD from the University of Oregon, Eugene. His research interests include security, privacy, databases, and distributed systems. He has authored or coauthored seven books, edited 52 books and conference proceedings, and published more than 500 technical papers in the refereed journals and conference proceedings. Five of his books have been translated in Chinese. He is also a holder of 23 patents. His current research sponsors are the Army Research Office, Office of Naval Research, National Security Agency, National Science Foundation, Northrop Grumman Corporation, and Intelligent Automation, Inc. Dr. Jajodia was elected a fellow of IEEE in January, 2013. He received the 1996 IFIP TC 11 Kristian Beckman award, 2000 Volgenau School of Engineering Outstanding Research Faculty Award, 2008 ACM SIGSAC Outstanding Contributions Award, 2011 IFIP WG 11.3 Outstanding Research Contributions Award, 2015 ESORICS Outstanding Research Award, 2016 Federal Information Systems Security Educators Association (FISSEA) Educator of the Year Award, 2016 IEEE Computer Society Technical Achievement Award, and 2020 IEEE Computer Society W. Wallace McDowell Award. He was recognized for the most accepted papers at the 30th anniversary of the IEEE Symposium on Security and Privacy. His h-index is 105 and Erdos number is 2. The URL for his web page is http://csis.gmu.edu/jajodia.

V.S. Subrahmanian is the Dartmouth College Distinguished Professor in Cybersecurity, Technology, and Society and Director of the Institute for Security, Technology, and Society at Dartmouth. He previously served as a Professor of Computer Science at the University of Maryland from 1989-2017 where he created and headed both the Lab for Computational Cultural Dynamics and the Center for Digital International Government. He also served for 6+ years as Director of the University of Marylands Institute for Advanced Computer Studies. Prof. Subrahmanian is an expert on big data analytics including methods to analyze text/geospatial/relational/social network data, learn behavioral models from the data, forecast actions, and influence behaviors with applications to cybersecurity and counterterrorism. He has written five books, edited ten, and published over 300 refereed articles. He is a Fellow of the American Association for the Advancement of Science and the Association for the Advancement of Artificial Intelligence and received numerous other honors and awards. His work has been featured in numerous outlets such as the Baltimore Sun, the Economist, Science, Nature, the Washington Post, American Public Media. He serves on the editorial boards of numerous journals including Science, the Board of Directors of the Development Gateway Foundation (set up by the World Bank), SentiMetrix, Inc., and on the Research Advisory Board of Tata Consultancy Services. He previously served on DARPAs Executive Advisory Council on Advanced Logistics and as an ad-hoc member of the US Air Force Science Advisory Board. Homepage: http://home.cs.dartmouth.edu/ vs/