Encrypted Authorship Attribution for Fraud Detection

Zita Baronnet[1][0009−0002−8224−9805] and Leonard Schild[2][0009−0007−3336−1258]

[1] Universität Trier, Trier 54286, Germany
lastname@uni-trier.de
[2] COSIC KU Leuven, Leuven 3000, Belgium
firstname.lastname@kuleuven.be

Abstract. Authorship attribution techniques are routinely used to relate anonymous or disputed texts to known authors through characteristic writing styles. Within the financial sector, such techniques may support investigations of phishing campaigns, impersonation attempts, investment scams, or other forms of fraud. More concretely, these methods can be leveraged to determine whether newly observed communications are linked to previously identified actors. In practice, however, both the communications under investigation and the intelligence databases used for attribution may contain sensitive information that cannot be shared across organizational boundaries.

In this work, we investigate the privacy-preserving evaluation of stylometric authorship attribution through fully homomorphic encryption (FHE). We focus on Burrows' Delta, a widely studied authorship attribution method based on token frequency statistics and z-score comparisons. We present a homomorphic implementation of Burrows' Delta, describe a motivating fraud-analysis scenario together with its threat model and trust assumptions, and then analyze the resulting computational complexity. Finally, we evaluate the practicality of the proposed approach using the CKKS homomorphic encryption scheme and show that the complete pipeline can be executed without bootstrapping while maintaining high numerical accuracy. Our results demonstrate that stylometric authorship attribution constitutes a realistic target for privacy-preserving outsourced analytics and illustrate how homomorphic encryption can support confidential fraud investigations.

Keywords: Homomorphic Encryption · Stylometry · Secure Computation

1 Introduction

Financial institutions routinely process large volumes of textual information ranging from legitimate customer communications to (spear-)phishing attempts sent to their employees. During investigations of phishing campaigns or any suspicious communications, one important question is whether multiple communications originate from the same actor despite being transmitted through dif-

ferent accounts, identities, or communication channels. Establishing such links can provide valuable evidence for attribution and fraud analysis.

Authorship attribution techniques address this problem by analyzing characteristic properties of an author's writing style. These methods fall within the broader field of stylometry [25] and have been studied extensively within computational linguistics [35] and digital forensics. Beyond their traditional applications, stylometric methods have also found use in financial and economic research [20] [18], though often under a range of different names; accounting and finance researchers use a range of terminology to describe stylometric research, such as textual analysis, computational linguistics, natural language processing, information retrieval and content analysis [28]. Examples of such applications include the analysis of financial reports [2], fraudulent documents [29], and other forms of textual data [41] [9]. Among the most influential approaches is Burrows' Delta [10], which is a distance-based method that compares standardized term-frequency statistics between documents and known author profiles. Burrows' Delta is computationally appealing because it relies primarily on counting, normalization, and vector-distance computations. Recently, Burrows' Delta was used on the anonymous minutes of the board meetings of Sweden's Central Bank between 2007 and 2009 to identify its speakers [3].

Although authorship attribution may be considered to be useful evidence for fraud detection, the underlying communications are often highly sensitive. Fraud investigations may involve customer information, transaction details, internal reports, or legally protected communications. At the same time, authorship attribution may rely on proprietary databases leading to a situation in which neither the document owner nor the analytics provider may be willing to disclose their data to the other party. Yet, homomorphic encryption can provide a compelling solution to this problem.

Homomorphic encryption schemes enable computations to be performed directly on encrypted data. The first homomorphic schemes were limited in the type of operations that they can carry out: the RSA scheme [42] and Pailler cryptosystem [37] respectively enabled multiplication and addition of encrypted values. Yet, the first fully homomorphic encryption scheme was only proposed in 2009 by Gentry [22]. Unlike prior attempts, the cryptosystem put limits neither on the type of computation carried out nor on the amount of calculations. Recent advances in homomorphic encryption have transformed the technology from a primarily theoretical concept into a practical tool for selected privacy-preserving analytics tasks and are featured in popular products [23]. Nevertheless, the efficient execution of complex natural-language-processing pipelines remains prohibitively expensive [4], motivating the search for domain-specific methods that can provide useful functionality while remaining compatible with the computational constraints of current FHE schemes.

Contribution In this work, we investigate the homomorphic evaluation of Burrows' Delta. We show that the structure of the algorithm leads to an efficient implementation based on token counting and low-degree arithmetic operations. In the considered parameter regime, this allows us to avoid the need for boot-

strapping, a special procedure that is considered a bottleneck computation in fully homomorphic schemes. We revisit the notion of bootstrapping later on. Our contributions are as follows:

1. We introduce a privacy-preserving authorship attribution pipeline for a fraud-analysis scenario.
2. We describe a homomorphic realization of Burrows' Delta and identify the components required for efficient encrypted evaluation
3. We give a thorough complexity analysis of our pipeline and demonstrate that the resulting circuit does not require bootstrapping.
4. We provide an implementation and evaluation based on OpenFHE [5].

We proceed as follows. In Section 2, we introduce the notation used throughout this work, followed by the necessary background regarding homomorphic encryption and stylometry. Afterwards, we describe our motivating scenario in further detail in Section 3, including our threat model and trust assumptions. Then, we describe a homomorphic version of Burrows' Delta in Section 4. Finally, we evaluate the pipeline in Section 5, including a practical evaluation and complexity analysis, before presenting a brief conclusion in Section 6.

2 Preliminaries

2.1 Notation

Let $\mathbb{Z}$ and $\mathbb{R}$ respectively denote the set of integers and reals respectively. Given $a \in \mathbb{Z}$, the set $[a]$ denotes $[a] = \{0, 1, ..., a-1\}$. For any $m \in \mathbb{Z}$, let $\mathbb{Z}_m$ denote the ring of integers modulo m, identified with the set $\{0, ..., m-1\}$. We write vectors in boldface, e.g. $\boldsymbol{x} \in \mathbb{R}^N$ for a dimension N. Finally, all other sets introduced are written in blackboard font e.g. a set of authors $\mathbb{A}$ and data-structure type varaibles in sans-serif font e.g. a document D.

2.2 Homomorphic Encryption

Fully homomorphic encryption schemes operate similarly to conventional symmetric or asymmetric cryptosystems, but are augmented with a set of procedures, enabling operations on the encrypted data. The first construction dates back to Gentry [22] but was deemed impractical in terms of computational complexity. Since then, efficient novel schemes were designed and used in practice [23]. Notable schemes include but are not limited to TFHE [13], FHEW [17], FINAL [7], BGV/BFV [8][19] and CKKS [12].

In this work, we rely on the CKKS scheme of which we present the high-level interface below. Note that we choose CKKS as Burrows' Delta ultimately relies on floating-point style operations, such as normalization, which can be carried out more easily within the CKKS compared to other FHE schemes.

- $\texttt{KeyGen}(\lambda) \rightarrow (\mathsf{PK}, \mathsf{SK}, \mathsf{EVK})$:
  Given a security parameter λ, returns a public key PK, secret key SK and evaluation key EVK.

- $\texttt{Encrypt}(\mathsf{PK}, \boldsymbol{x}) \to \mathsf{ct}$:
  Given a public key PK and datum $\boldsymbol{x} \in \mathbb{R}^N$, returns a ciphertext $\mathsf{ct} = \mathsf{Enc}(\boldsymbol{x})$.
- $\texttt{Decrypt}(\mathsf{SK}, \mathsf{ct}) \to \boldsymbol{x}$:
  Given a secret key SK and a ciphertext $\mathsf{ct} = \mathsf{Enc}(\boldsymbol{x})$, returns the message $\boldsymbol{x}$.
- $\texttt{Add}(\mathsf{EVK}, \mathsf{ct}_0, \mathsf{ct}_1) \to \mathsf{ct}^+$:
  Given an evaluation key EVK and ciphertexts $\mathsf{ct}_0 = \mathsf{Enc}(x_0), \mathsf{ct}_1 = \mathsf{Enc}(x_1)$ returns a ciphertext $\mathsf{ct}^+ = \mathsf{Enc}(\boldsymbol{x}_0 + \boldsymbol{x}_1)$.
- $\texttt{Mult}(\mathsf{EVK}, \mathsf{ct}_0, \mathsf{ct}_1) \to \mathsf{ct}^\times$:
  Given an evaluation key EVK and ciphertexts $\mathsf{ct}_0 = \mathsf{Enc}(x_0), \mathsf{ct}_1 = \mathsf{Enc}(x_1)$ returns a ciphertext $\mathsf{ct}^\times = \mathsf{Enc}(\boldsymbol{x}_0 \boxdot \boldsymbol{x}_1)$, where $\boxdot$ is the Hadamard or element-wise product.
- $\texttt{Rotate}(\mathsf{EVK}, \mathsf{ct}, k) \to \mathsf{ct}^\pi$: Given an evaluation key EVK, a ciphertext $\mathsf{ct} = \mathsf{Enc}(\boldsymbol{x})$ and rotation index k, returns a ciphertext $\mathsf{ct}^\pi = \mathsf{Enc}(\boldsymbol{x}^\pi)$ such that $\boldsymbol{x}^\pi_i = \boldsymbol{x}_{i+k \pmod N}$.

Throughout this work, we omit the evaluation key EVK from signatures and substitute `Add` (resp. `Mult`) by $+$ (resp. $\cdot$).

We note that all fully homomorphic encryption schemes rely on the hardness of lattice problems in order to achieve their security guarantees. More concretely, ciphertexts consists of (Module-)LWE [40] samples, which encrypt a value by introducing noise. During the evaluation of a circuit, noise terms are combined (e.g. during addition of values) and may reach a magnitude which "destroys" the encoded message, i.e. makes decryption fail. Therefore, all fully homomorphic schemes feature a so-called bootstrapping procedure, which aims to reduce the noise magnitude in order to enable further computation. Independently of the scheme, the bootstrapping procedure is considered the primary computational bottleneck. We do not discuss bootstrapping further as the low multiplicative depth of our approach does not require bootstrapping.

2.3 Stylometry

Stylometry developed into an academic discipline in the nineteenth century, and was first used as a term by Lutosławski who used it to describe his method for determining the chronology of Plato's *Dialogues* in 1897 [30]. His method consisted of analyzing five hundred peculiarities in Plato's *Dialogues* to study how Plato's writing style changed over time [30]. However, stylometry has a longer history, such as Mendehall's research on word length frequency distribution, and discourse among early Shakespearean scholars on authorship attribution, notably Edmond Malone [31]; [32].

Prior to the mid-twentieth century, stylometric research was conducted through traditional, non-digital methods [44]. This changed in 1965 when Mosteller and Wallace analyzed function words or 'style markers' using computer-assisted methods [48]. They analyzed function words in *The Federalist Papers* using the Bayes'

Theorem [34]. To determine whether Hamilton or Madison wrote the twelve disputed essays, they compared the frequency distribution rate per 1000 words of thirty function words [34]. The words 'from' and 'to' appear more frequently in Madison's than Hamilton's works, while Hamilton used the word 'by' more than Madison [34]. The results revealed that Madison most likely wrote the twelve disputed papers except possibly for Federalist No. 55 [34].

Mosteller and Wallace's study inspired many other researchers to use computer-assisted methods to conduct stylometry, launching a trend in non-traditional authorship attribution studies [48]. One such example was the study conducted by O'Brien and Darnell in which they analyzed the function words in various economic writings, one from the seventeenth century and five from the nineteenth century, to determine their authorship [35]. For example, they analyzed the function words in the seventeenth century pro-East India company pamphlet 'A Treatise Wherein is Demonstrated' to determine whether the author 'Philopatris' was the pseudonym of Sir Josiah Child [35].

By the late 1990s, more than three hundred non-traditional authorship attribution studies were published using various computer-assisted methods [44]. However, no one had found the definitive method for authorship attribution yet [44]. Rudman [44] criticized the many studies for their flawed experimental designs, inadequate due diligence, misapplication of statistical methods, poor quality training data, insufficient knowledge, and lack of transparency. Stamatatos [48] adds that most of these methods were computer-assisted not computer-based, meaning that their methods suffered from issues related to human error such as using too long textual data, too small number of candidate authors and ad hoc approaches [48]. In the early 2000s, researchers began to employ methods from information retrieval, machine learning and natural language processing (NLP), such as multiclass classification [48]. This set the stage for John Burrows to introduce his Delta method in 2001 [11].

2.4 Burrows' Delta

Assume we are given a document D of unknown authorship, which we treat as a vector of tokens $\mathsf{D} = [t_0, t_1, ..., t_{N-1}]$, and a set of candidate authors $\mathbb{A} = \{\mathcal{A}_0, \mathcal{A}_1, ..., \mathcal{A}_{k-1}\}$. We assume that $t_i \in \mathbb{T}$ where $\mathbb{T}$ is the set of all possible tokens, which we identify with the set $\mathbb{Z}_m$ with $m = |\mathbb{T}|$. We aim to determine the author $\mathcal{A}^* \in \mathbb{A}$ that is most likely to have produced the document D.

One of the most well-known techniques to infer authorship is Burrows' Delta. We describe the general version below and refer to [45] for further details. Burrows' Delta falls into the category of distance based stylometrical tests, in which documents are mapped to vectors and compared to vectors stemming from known authors. Generally, the vector dimension depends on the number of unique tokens, which can be significant[3]. Hence, for further processing, it is commonplace to only focus on chosen subset $\mathbb{T}' \subset \mathbb{T}$ of size m.

[3] E.g. the English language consists of at least 10^5 words. Though the concrete number is subject to debate [36].

Then, for each author $\mathcal{A}_i \in \mathbb{A}$, prior knowledge is provided in the form of z-score tables or vectors $\mathsf{Z}_i \in \mathbb{R}^m$ such that

$$\mathsf{Z}_{i,t_j} = \frac{h_{i,t_j} - \mu_{t_j}}{\sigma_{t_j}}$$

where h_{i,t_j} is the relative term frequency of token t_j across documents from author $\mathcal{A}_i$ that is:

$$h_{i,t_j} = \frac{\#\text{Times } t_j \text{ appears}}{\#\text{Times tokens from } \mathbb{T}' \text{ appear}}.$$

Note that h_{i,t_j} is calculated over all works from $\mathcal{A}_i$. Furthermore, μ_{t_j}, σ_{t_j} are the mean and standard deviation of the frequency of appearance of the token t_j across all authors and respective documents.

Next, we calculate the z-score table Z' for D, focusing only on tokens in $\mathbb{T}'$, and for each author the delta value $\delta_{\mathcal{A}_i,\mathsf{D}}$ is calculated as

$$\delta_{\mathcal{A}_i,\mathsf{D}} = \frac{1}{m}||\mathsf{Z}_i - \mathsf{Z}'||_1. \tag{1}$$

Note that Equation 1 applies the L1 norm, but other vector based similarity measures have been employed, e.g. the Euclidean norm or cosine similarity between $\mathsf{Z}_i, \mathsf{Z}'$. Finally, the classifier selects the most likely author as

$$\mathcal{A}^* = \arg\min_{\mathcal{A}\in\mathbb{A}} \delta_{\mathcal{A},\mathsf{D}}.$$

3 Application and Security Model

In this section, we present our motivating application scenario together with the threat model and trust assumptions that we shall follow throughout the remainder of this paper.

3.1 Motivation

We consider a setting in which a financial institution continuously, or on demand, investigates suspicious communications related to phishing campaigns, investment scams, impersonation attempts, and other forms of financial fraud. One objective of such investigations is to determine whether newly observed communications originate from a known set of malicious actors, even when these actors employ different identities, accounts, domains, or communication channels. Authorship attribution techniques provide a mechanism for linking anonymous or disputed texts through characteristic linguistic features and writing styles. Similar approaches have previously been investigated in the context of email analysis, phishing mitigation, and forensic authorship attribution [49].

To examine this issue, consider the following scenario: a financial institution receives a newly observed suspicious communication and wishes to determine

whether it is stylistically related to known fraud campaigns. To this end, a specialized fraud analysis provider maintains a collection of stylometric profiles derived from previously identified fraud actors and campaigns. These profiles represent valuable intelligence accumulated over time and may themselves constitute proprietary or otherwise sensitive information. At the same time, the communication under investigation may contain customer information, transaction details, internal investigative material, or other confidential data that cannot be disclosed to external parties. Consequently, neither the financial institution nor the fraud analysis provider may be willing to reveal their respective information assets to the other party.

Therefore, we propose an outsourced analytics setting in which the institution encrypts a document prior to submission. The analysis provider evaluates a stylometric test, specifically Burrows' Delta, directly on the encrypted document using its locally maintained stylometric profiles. The resulting authorship scores remain encrypted throughout the computation and are returned to the institution for decryption and interpretation. During the evaluation process, the contents of the investigated communication remain hidden from the analysis provider while the provider's stylometric intelligence remains undisclosed to the institution. Although we focus on a single financial institution interacting with a specialized analysis provider, the same architecture naturally extends to collaborative fraud detection settings in which multiple institutions contribute historical fraud corpora or stylometric profiles without exposing ongoing investigations or sensitive communications.

3.2 Threat Model and Leakage

The system consists of three parties:

1. A financial institution responsible for collecting suspicious communications and encrypting them prior to analysis.
2. An analysis provider that performs the stylometric evaluation.
3. A designated recipient authorized to decrypt and interpret the resulting authorship scores. Depending on the deployment scenario, this party may coincide with the financial institution.

The financial institution possesses a confidential document D, whose authorship is to be evaluated. Prior to outsourcing the computation, the institution encrypts the document under the CKKS scheme and transmits the resulting ciphertexts to the analysis provider. The analysis provider maintains a collection of stylometric profiles and the public parameters required to execute Burrows' Delta, but does not possess the decryption key.

We assume an honest-but-curious analysis provider that correctly executes the prescribed protocol, but may attempt to infer information about the submitted document from the ciphertexts, and any auxiliary information available during the computation. Hence, the primary objective of the protocol is to prevent

disclosure of the document contents while still enabling authorship determination. Furthermore, we assume that the designated recipient correctly performs decryption and obtains access to the final classification result. Therefore, any information revealed by the decrypted authorship scores is regarded as intentional functionality rather than protocol leakage. For the sake of simplicity, we do not consider malicious deviations from the protocol in the present work. Verifiable execution mechanisms based on zero-knowledge techniques could be incorporated to provide guarantees regarding correct computation, but would introduce significant additional complexity and are left for future work.

To close this section, we note that there remain several sources of leakage that even honest-but-curious parties may try to leverage. More specifically, the size of the encrypted document correlates to the size of the cleartext document. Therefore, certain conclusion could be drawn by an observer based on the concrete size of the ciphertexts. Furthermore, if stylometryic profiles are provided by a third party as we suggest, a synchronization step must occur between the encrypting party and the third party. This step is necessary as all parties must agree to the same encoding strategy, which we discuss further in Section 4. In this case, the encoding parameters utilized may leak information regarding language, but also specific attributes included in the stylometric profiles. However, such leakage is not inherent to our encrypted pipeline, and arises similarly if different privacy-preserving computation technologies are exploited.

3.3 Trust Assumptions

The confidentiality guarantees of the proposed construction are derived solely from the semantic security of the underlying homomorphic encryption scheme. In particular, we do not assume the presence of trusted execution environments, secure enclaves, trusted hardware, or remote-attestation infrastructures.

Depending on deployment requirements, the decryption capability may additionally be distributed among several parties using threshold homomorphic encryption. Such extensions could provide additional organizational oversight, e.g. by requiring that multiple parties jointly authorize access to the final result. Since these mechanisms are orthogonal to the encrypted evaluation of Burrows' Delta, they are outside the scope of the present work. A comparison between the proposed approach and trusted-hardware-based alternatives is provided in Section 5.

4 Methods

In this section, we will discuss the steps necessary to enable the homomorphic evaluation of Burrows' Delta.

First, we must focus on the encoding strategy of the document. As previously mentioned, we identify each token t_i with an element of $\mathbb{Z}_m$, which suggests encoding D directly as vectors of elements over $\mathbb{Z}_m$. However, the first step of Burrows' Delta consists of calculating the relative frequency of each token, which

in turn requires counting the amount of times each one appears. It is possible to perform a direct comparison of values within the CKKS scheme via specialized methods [26] or using special encodings that are maintained during the entire circuit evaluation, which have been investigated in [21].

While a direct arithmetic encoding of token identifiers would permit the use of existing CKKS comparison techniques, such methods require polynomial approximations of discontinuous functions whose accuracy depends on the magnitude and scaling of the compared values. Since token identifiers may span a substantially larger numeric range than the frequencies ultimately used by Burrows' Delta, parameter selection becomes closely tied to the chosen vocabulary representation. Instead, we employ a bitwise encoding of tokens. Although this does not necessarily reduce multiplicative depth, impacting practical performance, it yields circuits whose depth depends only on the token bit-width and whose precision requirements are largely determined by token frequencies occurring in the analyzed documents.

More concretely, we then represent each token t_i as a vector of its bits $\boldsymbol{t}_i \in \{0,1\}^w$ such that $t_i = \sum_{j=0}^{w-1} t_{i,j} 2^j, t_{i,j} \in \{0,1\}$ and encrypt

$$\mathsf{D} = [t_{0,0}, t_{0,1}, ..., t_{1,0}, ..., t_{n-1,w-1}].$$

For the sake of simplicity, we assume that the bit-width w is a power of two $w = 2^r$. In this form, comparison and counting become easy as they reduce to multiplications. Specifically, comparison of bits reduces to an XOR gate or $(b_0 - b_1)^2$ over $\mathbb{R}$ whereas the comparison of two bit-sequences $b, b' \in \{0.0, 1.0\}^w$ can be expressed as

$$\prod_{i=0}^{w-1} (1 - (b_i - b'_i)^2) = \begin{cases} 1 & \text{if } b = b' \\ 0 & \text{else} \end{cases} \tag{2}$$

Unlike in [21], we do not need to convert the encrypted comparison bit to a suitable domain as the comparison was evaluated over $\mathbb{R}$. Instead, note that after comparing a vector $\mathsf{D}' = [t_{0,0}, ..., t_{1,0}, ..., t_{n-1,w-1}]$ to a token $\bar{\boldsymbol{t}} = [\bar{t}_0, \bar{t}_1, ..., \bar{t}_{w-1}]$ through Equation 2 we obtain a vector of binary values $\boldsymbol{c}_{\mathsf{cmp}} = [c_0, 0, ..., c_1, 0, ...]$ such that $c_i = 1$ if and only if $\forall i : t_{0,i} = \bar{t}_{0,i}$. Hence, obtaining the number of times $\bar{t}$ appears reduces to summing every value in the comparison vector $\boldsymbol{c}_{\mathsf{cmp}}$. The overall procedure is given in detail in Algorithm 1. In the same algorithm, the bitwise comparison is given in lines 6,7 after which it is necessary to evaluate the product as in Equation 2. This step is carried out in line 11 to line 13, which implement a binary reduction tree. Equivalent reductions appear throughout this work to compute the component-wise sum or product of vectors, as the binary reduction tree approach induces an operation depth logarithmic in the length or dimension of the vector.

Once every token in $\mathbb{T}'$ has been counted, we must calculate the corresponding z-score vector. The procedure is given in Algorithm 2. The main challenge of this step is the normalization required to obtain the relative frequency scores. Performing the normalization depends on calculating the reciprocal of the total

Algorithm 1: `CountToken`

Input: Ciphertext $(\mathsf{ct}_i)_{i\in[l]} = \mathsf{Enc}(\boldsymbol{d}_i)$, $\boldsymbol{d}_i \in \mathbb{R}^N$ representing D
Input: Target Token t
Data: Bitwidth $w = 2^r$ for used for token representation
Output: Ciphertext $\mathsf{ct}_{\mathsf{out}}$ encrypting the absolute frequency of t in D

Set $(t_j)_{j\in[w]}$ s.t. $t = \sum_i t_j \cdot 2^j$;
Let $\boldsymbol{m} \in \mathbb{R}^N, \boldsymbol{m}_{a\cdot w+j} = t_j$;
Let $\boldsymbol{v} \in \mathbb{R}^N, \boldsymbol{v}_{a\cdot w+j} = 1$ if $j = 0$, else 0;
Let $\boldsymbol{c} \in \mathbb{R}^N, \boldsymbol{c}_i = 1$ if $i = 0$, else 0;
for $i = 0$ *to* $i = l - 1$ **do**
  $\mathsf{ct}'_i \leftarrow \mathsf{ct}_i - \boldsymbol{m}$;
  $\mathsf{ct}'_i \leftarrow \mathbf{1} - (\mathsf{ct}'_i)^2$;
end
$\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{Enc}(\mathbf{0})$;
for $i = 0$ *to* $i = l - 1$ **do**
  for $j = r - 1$ *to* $j = 0$ **do**
    $\mathsf{ct}^{\pi}_i \leftarrow \mathtt{Rotate}(\mathsf{ct}'_i, -2^j)$;
    $\mathsf{ct}'_i \leftarrow \mathsf{ct}'_i \cdot \mathsf{ct}^{\pi}_i$;
  end
  `/* Mask out values that are not at indices` j, $j = 0 \pmod w$ `*/`
  $\mathsf{ct}'_i \leftarrow \mathsf{ct}'_i \cdot \boldsymbol{v}$;
  for $j = \log_2(N) - 1$ *to* $j = r$ **do**
    $\mathsf{ct}^{\pi}_i \leftarrow \mathtt{Rotate}(\mathsf{ct}'_i, -2^j)$;
    $\mathsf{ct}'_i \leftarrow \mathsf{ct}'_i + \mathsf{ct}^{\pi}_i$;
  end
  `/* Mask to keep only term at index 0 */`
  $\mathsf{ct}'_i \leftarrow \mathsf{ct}'_i \cdot \boldsymbol{c}$;
  $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{ct}_{\mathsf{out}} + \mathsf{ct}'_i$;
end
Result: $\mathsf{ct}_{\mathsf{out}}$

number of times the tokens in $\mathbb{T}'$ appear. Although the CKKS scheme enables operations over $\mathbb{R}$ by internally representing values through a fixed point approximation, the homomorphic operations themselves are carried out over a finite field or module. Consequently, a direct calculation of the inverse is not possible, e.g. by inverting the encoded ciphertext. Instead, we rely on a polynomial approximation. The accuracy of such approximations depends on how precisely we can estimate the domain of values to be inverted. In our setting, we may determine bounds based on prior knowledge of the authors we aim to compare the document to. However, the inversion may be carried out via Goldschmidt iteration even in a setting where no such estimates are available, and refer to [33] for more details. Furthermore, we assume that in our application scenario a division by zero never occurs, i.e. at least one element of $\mathbb{T}'$ is present in the document.

Algorithm 2: `ConstructZScore`

Input: Ciphertext $(\mathsf{ct}_i)_{i\in[l]} = \mathsf{Enc}(\boldsymbol{d}_i)$, $\boldsymbol{d}_i \in \mathbb{R}^N$ representing D
Input: Token set $\mathbb{T}'$
Input: Mean vector $\boldsymbol{\mu} \in \mathbb{R}^m$
Input: Standard Deviation vector $\boldsymbol{\sigma} \in \mathbb{R}^m$
Data: Bitwidth $w = 2^r$ for used for token representation
Output: Ciphertext $\mathsf{ct}_{\mathsf{out}}$ encrypting the z-scores for D

$\mathsf{ct}_{\mathsf{counts}} \leftarrow \mathsf{Enc}(\mathbf{0})$;
$\mathsf{ct}_{\mathsf{sum}} \leftarrow \mathsf{Enc}(\mathbf{0})$;
for t_i *in* $\mathbb{T}'$ **do**
  $\mathsf{ct}'_i \leftarrow \mathtt{CountToken}((\mathsf{ct}_i)_{i\in[l]}, t_i)$;
  $\mathsf{ct}^{\pi}_i \leftarrow \mathtt{Rotate}(\mathsf{ct}'_i, i)$;
  $\mathsf{ct}_{\mathsf{sum}} \leftarrow \mathsf{ct}_{\mathsf{sum}} + \mathsf{ct}'_i$;
  $\mathsf{ct}_{\mathsf{counts}} \leftarrow \mathsf{ct}_{\mathsf{counts}} + \mathsf{ct}^{\pi}_i$;
end
$\mathsf{ct}_{\mathsf{inv}} \leftarrow \mathtt{ComputeInverse}(\mathsf{ct}_{\mathsf{sum}})$;
`/* Replicate to fill every slot with inverse value */`
for $i = \log_2(N) - 1$ *to* $i = 0$ **do**
  $\mathsf{ct}^{\pi}_{\mathsf{inv}} \leftarrow \mathtt{Rotate}(\mathsf{ct}_{\mathsf{inv}}, 2^i)$;
  $\mathsf{ct}_{\mathsf{inv}} \leftarrow \mathsf{ct}_{\mathsf{inv}} + \mathsf{ct}^{\pi}_{\mathsf{inv}}$;
end
$\mathsf{ct}_{\mathsf{rtf}} \leftarrow \mathsf{ct}_{\mathsf{counts}} \cdot \mathsf{ct}_{\mathsf{inv}}$;
$\mathsf{ct}_{\mathsf{zscore}} \leftarrow \frac{1}{\boldsymbol{\sigma}} \cdot (\mathsf{ct}_{\mathsf{rtf}} - \boldsymbol{\mu})$;
Result: $\mathsf{ct}_{\mathsf{zscore}}$

Finally, the homomorphic computation of delta scores is given in Algorithm 3. Looking back at Equation 1, we observe that most operations can be carried out immediately as they reduce to additions or multiplications. In order to compute the L1 norm, we must compute the absolute value, which we implement via the identity $|x| = \sqrt{x^2}$ and obtain the square root by polynomial approximation. The interval for such an approximation is obtained by first determining the range $[Z_{\min}, Z_{\max}]$ where $Z_{\min}, Z_{\max}$ are the minimum (resp. maximum) observed z-scores in the known z-score tables. Then, the approximation interval is obtained as $[0, \alpha(Z_{\max} - Z_{\min})^2]$ where $\alpha \geq 1$ is a "safety" factor though $\alpha = 1$ was sufficient in our experiments. The result of Algorithm 3 is a ciphertext ct_δ encrypting a vector $\boldsymbol{x}_\delta$ containing all delta scores and specifically $\boldsymbol{x}_{\delta,i} = \delta_{\mathcal{A}_i,\mathsf{D}}$. In practice, this is preferable over the minimum value or argument as scores may be close in value and no clear author can be identified. For the sake of completeness, we refer to [27] for a discussion of how to obtain the minimum argument.

5 Evaluation

In this section, we discuss the implementation and practical performance of the algorithms we described in Section 4.

Algorithm 3: `ComputeDeltaScores`

Input: Ciphertext $(\mathsf{ct}_i)_{i\in[l]} = \mathsf{Enc}(\boldsymbol{d}_i)$, $\boldsymbol{d}_i \in \mathbb{R}^N$ representing D
Input: Token set $\mathbb{T}'$
Input: Mean vector $\boldsymbol{\mu} \in \mathbb{R}^m$
Input: Standard Deviation vector $\boldsymbol{\sigma} \in \mathbb{R}^m$
Input: Z-score tables $(\mathsf{Z}_i)_{i\in[k]}$
Data: Bitwidth $w = 2^r$ for used for token representation
Output: Ciphertext $\mathsf{ct}_{\mathsf{out}}$ encrypting the delta scores

Let $\boldsymbol{c} \in \mathbb{R}^N, \boldsymbol{c}_i = \frac{1}{m}$ if $i = 0$, else 0;
$\mathsf{ct}_{\mathsf{zscore}} \leftarrow \mathtt{ConstructZScore}((\mathsf{ct}_i)_{i\in[l]}, \mathbb{T}', \boldsymbol{\mu}, \boldsymbol{\sigma})$;
$\mathsf{ct}_\delta \leftarrow \mathsf{Enc}(\mathbf{0})$;
for $i = 0$ *to* $i = k - 1$ **do**
  $\mathsf{ct}'_i \leftarrow \mathsf{ct}_{\mathsf{zscore}} - \mathsf{Z}_i$;
  $\mathsf{ct}'_i \leftarrow (\mathsf{ct}'_i)^2$;
  $\mathsf{ct}'_i \leftarrow \mathtt{ComputeSqrt}(\mathsf{ct}'_i)$;
  for $j = \log_2(N) - 1$ *to* $j = 0$ **do**
    $\mathsf{ct}_i^\pi \leftarrow \mathtt{Rotate}(\mathsf{ct}'_i, -2^j)$;
    $\mathsf{ct}'_i \leftarrow \mathsf{ct}'_i + \mathsf{ct}_i^\pi$;
  end
  `/* Mask to keep only term at index 0 */`
  $\mathsf{ct}'_i \leftarrow \mathsf{ct}'_i \cdot \boldsymbol{c}$;
  $\mathsf{ct}_i^\pi \leftarrow \mathtt{Rotate}(\mathsf{ct}'_i, i)$;
  $\mathsf{ct}_\delta \leftarrow \mathsf{ct}_\delta + \mathsf{ct}_i^\pi$;
end
Result: ct_δ

5.1 Practical Performance

We implement the algorithms[4] using the OpenFHE [5] library, compiled with clang and relying on Intel HEXL [6]. The code is executed on a machine with an Intel i7-14700KF and 64GB of RAM and run in parallel over 32 threads. We give a general performance overview in Figure 1. The multiplicative depth of each Algorithm is given in Table 2. As rotations and multiplications are considered to be the primary computational bottleneck for the CKKS scheme, we give the number of rotations and multiplication required for each algorithm in Table 3. Furthermore, the total depth, number of rotations and multiplications are given in Table 4. For the sake of completeness, we include the parameters of the CKKS scheme used in Table 1, and refer to [12] for a further discussion.

We observe in Figure 1 that independently of which parameter is varied, the impact on runtime is quasi-linear. This observation matches the design of Algorithm 1,2,3 and is expected. However, the concrete change in execution time still varies significantly by parameter. A change in the number of candidate authors $\mathbb{A}$ only becomes relevant in Algorithm 3. Hence, the work done in Algorithm 1 need not be repeated and larger $\mathbb{A}$ sizes only have a minor impact compared

[4] https://anonymous.4open.science/r/Homomorphic-Encryption-HE-Pipeline-for-Stylometry-823C/README.md

Table 1: CKKS and encoding parameters. λ is the security parameter, N' denotes the ring dimension used, Δ the scaling factor. $\mathsf{d_{inv}}, \mathsf{d_{sqrt}}$ correspond to the degrees of the polynomial approximation of inverse and square root function.

Parameter	w	λ	N'	$\log_2(\Delta)$	Circuit depth	$\mathsf{d_{inv}}, \mathsf{d_{sqrt}}$
Value	32	128	2^{17}	50	20	16

Table 2: Multiplicative depth required for each sub-procedure **excluding** calls to other algorithms. w is the bit-width of the token encoding, $\mathsf{d_{inv}}, \mathsf{d_{sqrt}}$ are the degree of polynomial approx. of the inverse and square root function.

	Algorithm 1	Algorithm 2	Algorithm 3
Depth	$\log_2(w) + 3$	$\log_2(\mathsf{d_{inv}}) + 1$	$\log_2(\mathsf{d_{sqrt}}) + 2$

to a change in the set of tokens considered $\mathbb{T}'$ (Figure 1a vs. Figure 1b). In the latter case, increasing $|\mathbb{T}'|$ requires additional iterations in Algorithm 2 and calls to Algorithm 1, which corresponds to the bottleneck in practice. Furthermore, a similar effect can be observed by increasing the size of the document (Figure 1c) as more ciphertexts are required to encode the document, which in turn need to be processed through Algorithm 1.

Next, we briefly compare the homomorphic evaluation to the cleartext one. On the one hand, we measure the relative error of the homomorphic evaluation compared to a cleartext implementation as accuracy is often a concern when calculations are carried out through HE. Across our experiments, we observe a worst-case relative error of 10^{-4}. If insufficient, smaller relative errors may be obtained by relying on higher-degree polynomial approximations for the inverse and square-root function, possibly at a loss in computational performance. On the other hand, we note that, even in the best case, in which we set $|\mathbb{A}| = 1$, the runtime drastically exceeds that of the cleartext equivalent which for all cases remains below one second.

Finally, note that for all parameter variations considered in this Section (e.g. $|\mathbb{T}'|$ or $|\mathbb{A}|$), the total multiplicative depth remains below the supported depth of the chosen CKKS parameterization (c.f. Table 1) and, therefore, no bootstrapping is required. Indeed, recall that w is the number of bits chosen for the token representation. Then, Table 4 highlights that the multiplicative depth of the pipeline itself is logarithmic in this parameter, yielding a very shallow circuit in general. This further justifies the use of Burrows' Delta over more advanced methods such as language based models (e.g [16]), whose homomorphic evaluation is particularly expensive, see [43].

Table 3: Operation count required for each sub-procedure **excluding** calls to other algorithms. N is the dimension of vectors encrypted in ciphertexts. l is the number of ciphertexts required to store the document, w is the bit-width of the token encoding, $\mathsf{d_{inv}}, \mathsf{d_{sqrt}}$ are the degree of polynomial approx. of the inverse and square root function.

	Algorithm 1	Algorithm 2	Algorithm 3
Rotations	$l \cdot \log_2(N)$	$\lvert\mathbb{T}'\rvert + \log_2(N)$	$\lvert\mathbb{A}\rvert \cdot (\log_2(N) + 1)$
Multiplications	$l \cdot (\log_2(w) + 2)$	$\sqrt{\mathsf{d_{inv}}} + 1$	$\lvert\mathbb{A}\rvert \cdot (2 + \sqrt{\mathsf{d_{sqrt}}})$

Table 4: Total depth and operation count for the pipeline. N is the dimension of vectors encrypted in ciphertexts. l is the number of ciphertexts required to store the document, w is the bit-width of the token encoding, $\mathsf{d_{inv}}, \mathsf{d_{sqrt}}$ are the degree of polynomial approx. of the inverse and square root function.

Depth.	Multiplications	Rotations
$\log_2(w) + \log_2(\mathsf{d_{inv}}) + \log_2(\mathsf{d_{sqrt}}) + 5$	$\lvert\mathbb{T}'\rvert \cdot l \cdot (\log_2(w) + 2) + \sqrt{\mathsf{d_{inv}}} + 1 + \lvert\mathbb{A}\rvert \cdot (2 + \sqrt{\mathsf{d_{sqrt}}})$	$\lvert\mathbb{T}'\rvert \cdot (1 + l \cdot \log_2(N)) + \log_2(N) + \lvert\mathbb{A}\rvert \cdot (\log_2(N) + 1)$

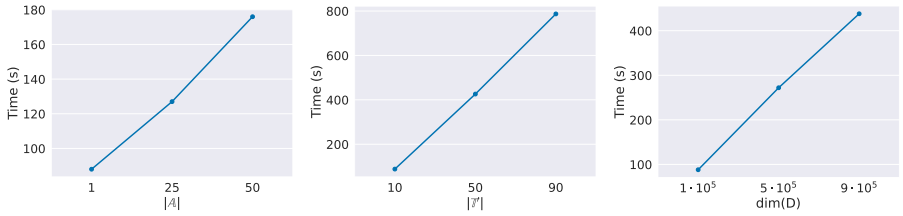
5.2 Comparison with Trusted Execution Environments

An alternative technology for confidential computation consists of trusted execution environments (TEEs). Conceptually, TEEs rely on hardware attestation to establish protected execution environments in which even the owner of the hosting system is prevented from directly observing the executed computation. Prominent examples include Intel SGX [15], Intel TDX [24], and AMD Secure Encrypted Virtualization (SEV) [1]. The workflow described in Section 3 could, in principle, be realized using a TEE instead of homomorphic encryption.

In such a deployment, once remote attestation of an enclave succeeds, the document would be transferred to the enclave, either via secure channel or encrypting it using a device depended public key. Then, once the document becomes available inside the enclave, the stylometric analysis would be performed in plaintext, and the resulting scores would subsequently be encrypted before leaving the trusted environment. TEE-based approaches generally offer significantly lower computational overhead than homomorphic encryption. However, they require trust in additional infrastructure, including the hardware manufacturer, attestation mechanisms, firmware updates, and the security of the enclave implementation itself. Furthermore, TEEs have repeatedly been shown vulnerable to both hardware- and software-based attacks [46, 14, 47]. As a result, confidentiality ultimately depends not only on cryptographic assumptions of the public key scheme or secure channel but also on the continued security of a particular hardware platform.

In contrast, homomorphic encryption eliminates the need for trusted hardware during computation. The analysis server operates exclusively on ciphertexts

Fig. 1: Change in runtime depending on parameter variation. Time is given in seconds.



(a) Execution time depending on $|\mathbb{A}|$. $|\mathbb{T}'| = 10, \dim(\mathsf{D}) = 10^5$

(b) Execution time depending on $|\mathbb{T}'|$. $|\mathbb{A}| = 1, \dim(\mathsf{D}) = 10^5$

(c) Execution time depending on $\dim(\mathsf{D})$. $|\mathbb{T}'| = 10, |\mathbb{A}| = 1$

and never obtains access to plaintext documents or intermediate stylometric features. The current FHE schemes rely on lattice-based hardness assumptions closely related to those employed by recently standardized post-quantum cryptographic schemes, such as ML-KEM and ML-DSA [39, 38]. Consequently, the confidentiality of the analyzed documents is independent of the execution hardware and remains protected even if the evaluation server is fully compromised. This distinction is particularly attractive in collaborative fraud-detection scenarios involving multiple organizations. In such settings, participants may be unwilling to rely on a common hardware vendor or to provision sensitive communications into a third-party enclave. Homomorphic encryption provides an alternative trust model in which the computation provider need not be trusted with access to the underlying documents. Therefore, we view TEEs and homomorphic encryption as complementary technologies; TEEs offer superior performance, whereas homomorphic encryption minimizes trust in the execution environment and provides stronger confidentiality guarantees.

6 Conclusion and Future Work

In this work, we investigated and discussed the feasibility of encrypted authorship attribution for fraud detection. However, we leave the investigation of other computation-on-encrypted-data technologies to future work, e.g. multi-party computation or verifiable HE.

References

1. Advanced Micro Devices, Inc.: Amd secure encrypted virtualization (sev). https://www.amd.com/en/developer/sev.html (2023), accessed: 2026-09-25
2. Alduais, F., Almasria, N.A., Samara, A., Masadeh, A.: Conciseness, financial disclosure, and market reaction: A textual analysis of annual reports in listed chinese companies. International Journal Financial Studies **10**(4), 1–22 (2022). https://doi.org/10.3390/ijfs10040104

3. Alumäe, T., Fishel, M.: Who said what? speaker identification from anonymous minutes of meetings. In: Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa). pp. 124–134. University of Tartu Library (2023)
4. Badawi, A.A., Alexandru, A., Polyakov, Y., Vaikuntanathan, V.: SoK: Private LLM inference using approximate homomorphic encryption. Cryptology ePrint Archive, Paper 2026/935 (2026), https://eprint.iacr.org/2026/935
5. Badawi, A.A., Bates, J., Bergamaschi, F., Cousins, D.B., Erabelli, S., Genise, N., Halevi, S., Hunt, H., Kim, A., Lee, Y., Liu, Z., Micciancio, D., Quah, I., Polyakov, Y., R.V., S., Rohloff, K., Saylor, J., Suponitsky, D., Triplett, M., Vaikuntanathan, V., Zucca, V.: Openfhe: Open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing and Applied Homomorphic Cryptography. pp. 53–63 (2022). https://doi.org/10.1145/3560827.3563379
6. Boemer, F., Kim, S., Seifu, G., de Souza, F.D., Gopal, V., et al.: Intel homomorphic encryption (he) acceleration library. Intel HEXL (release 1.2) (2021), https://intellabs.github.io/hexl/doxygen/html/index.html
7. Bonte, C., Iliashenko, I.I., Park, J., Pereira, H.V.L., Smart, N.P.: Final: Faster fhe instantiated with ntru and lwe. In: Advances in Cryptology – ASIACRYPT 2022. vol. 13792, p. 188–215. Springer Nature (2022). https://doi.org/10.1007/978-3-031-22966-4_7
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS '12: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 309–325. Association for Computing Machinery (ACM) (2012). https://doi.org/10.1145/2090236.209026
9. Bua, G., Kapp, D., Ramella, F., Rognone, L.: Transition versus physical climate risk pricing in european financial markets: a text-based approach. The European Journal of Finance **30**(17), 2076–2110 (2024). https://doi.org/10.1080/1351847X.2024.2355103
10. Burrows, J.: 'delta': a measure of stylistic difference and a guide to likely authorship. Literary and Linguistic Computing **17**(3), 267–287 (2002). https://doi.org/10.1093/llc/17.3.267
11. Burrows, J.: Questions of authorship: Attribution and beyond: A lecture delivered on the occasion of the roberto busa award ach-allc 2001, new york. Computers and the Humanities **37**(1), 5–32 (2003). https://doi.org/https://www.jstor.org/stable/30204877
12. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology - ASIACRYPT 2017. vol. 10624, pp. 409–437. Springer Nature (2017). https://doi.org/10.1007/978-3-319-70694-8_15
13. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology - ASIACRYPT 2016. vol. 10031, pp. 3–33. Springer (2016). https://doi.org/10.1007/978-3-662-53887-6_1
14. Chuang, J., Alex Seto, N.B., Stephan van Schaik, C.G., Genkin, D.: Tee.fail: Breaking trusted execution environments via ddr5 memory bus interposition. In: 2026 IEEE Symposium on Security and Privacy (SP). pp. 4408–4426. IEEE (2026), https://ieeexplore.ieee.org/document/11573449
15. Costan, V., Devadas, S.: Intel sgx explained. IACR Cryptology ePrint Archive **2016**, 86 (2016), https://eprint.iacr.org/2016/086

16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019), https://arxiv.org/abs/1810.04805
17. Ducas, L., Micciancio, D.: Fhew: Bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology - EUROCRYPT 2015. vol. 9057, pp. 617–640. Springer Nature (2015). https://doi.org/10.1007/978-3-662-46800-5_24
18. Fama, E.F., French, K.R.: Common risk factors in the returns on stocks and bonds. Journal of Financial Economics **33**(1), 3–56 (1993). https://doi.org/10.1016/0304-405X(93)90023-5
19. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
20. Frazier, K.B., Ingram, R.W., Tennyson, B.M.: A methodology for the analysis of narrative accounting disclosures. Journal of Accounting Research **22**(1), 318–331 (1984). https://doi.org/10.2307/2490713
21. Gao, M., Zheng, H.: Fhe for simd arithmetic logic units with amortized o(1) bootstrapping per ciphertext. In: Advances in Cryptology - CRYPTO 2026. pp. 299–330. Springer Nature Switzerland (2026). https://doi.org/10.1007/978-3-032-35374-0_10
22. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC '09: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178. Association for Computing Machinery (ACM) (2009). https://doi.org/10.1145/1536414.1536440
23. Inc., A.: Combining machine learning and homomorphic encryption in the apple ecosystem. Apple Machine Learning Research (2024), https://machinelearning.apple.com/research/homomorphic-encryption
24. Intel Corporation: Architecture specification: Intel trust domain extensions (intel tdx) module. Tech. Rep. 344425-005US, Intel Corporation (2023), https://cdrdv2-public.intel.com/733568/tdx-module-1.0-public-spec-344425005.pdf
25. Iqbal, F., Binsalleeh, H., Fung, B.C., Debbab, M.: Mining writeprints from anonymous e-mails for forensic investigation. Digital Investigation **7**(1-2), 1–9 (2010). https://doi.org/10.1016/j.diin.2010.03.003
26. Lee, E., Lee, J.W., No, J.S., Kim, Y.S.: Minimax approximation of sign function by composite polynomial for homomorphic comparison. IEEE Transactions on Dependable and Secure Computing **19**(6), 3711–3727 (2022). https://doi.org/10.1109/TDSC.2021.3105111
27. Lorenzo Rovida, A.L., Basile, S.: Lightweight sorting in approximate homomorphic encryption. IACR Communications in Cryptology **3**(1), 1–29 (2026). https://doi.org/10.62056/a3ksdkmol
28. Loughran, T., McDonald, B.: Textual analysis in accounting and finance: A survey. Journal of Accounting Research **54**(4), 1187–1230 (2016). https://doi.org/10.1111/1475-679X.12123
29. Loughran, T., McDonald, B.: Textual analysis in finance. Annual Review of Financial Economics **12**, 357–375 (2020). https://doi.org/10.1146/annurev-financial-012820-032249
30. Lutosławski, W.: The Origin and Growth of Plato's Logic; With an Account of Plato's Style and of the Chronology of his Writings. Longmans, Green & Co. (1897), https://archive.org/details/originandgrowth00lutogoog/page/n8/mode/2up
31. Malone, E.: A Dissertation on the Three parts of King Henry VI. Tending to Shew That Those Plays Were not Written Originally by Shakspeare. Press

of Henry Baldwin (1787), https://archive.org/details/bim_eighteenth-century_a-dissertation-on-the-th_malone-edmond_1787/page/n1/mode/2up
32. Mendenhall, T.C.: The characteristic curves of composition. Science **9**(214), 237–249 (1887), https://www.jstor.org/stable/1764604
33. Moon, J., Omarov, Z., Yoo, D., An, Y., Chung, H.: Adaptive successive over-relaxation method for a faster iterative approximation of homomorphic operations. Cryptology ePrint Archive, Paper 2024/1366 (2024), https://eprint.iacr.org/2024/1366
34. Mosteller, F., Wallace, D.L.: Inference and Disputed Authorship: The Federalist. Addison-Wesley Publishing Company, Inc. (1964), https://archive.org/details/inferencedispute00most/page/n3/mode/2up
35. O'Brien, D., Darnell, A.: Authorship Puzzles in the History of Economics: A Statistical Approach. Macmillan Press Ltd (1982)
36. Oxford English Dictionary: Oxford english dictionary (2026), https://www.oed.com/, accessed: 2026-09-24
37. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology – EUROCRYPT '99. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999). https://doi.org/10.1007/3-540-48910-X_16, https://doi.org/10.1007/3-540-48910-X_16
38. Raimondo, G.M., Locascio, L.E.: Module-lattice-based digital signature standard. Tech. rep., National Institute of Standards and Technology (2024). https://doi.org/10.6028/NIST.FIPS.204
39. Raimondo, G.M., Locascio, L.E.: Module-lattice-based key-encapsulation mechanism standard. Tech. rep., National Institute of Standards and Technology (2024). https://doi.org/10.6028/NIST.FIPS.203
40. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56**(6), 1–40 (2009). https://doi.org/10.1145/1568318.1568324
41. Ren, R., Wu, D.D.: Forecasting stock market movement direction using sentiment analysis and support vector machine. IEEE Systems Journal **13**(1), 760–770 (2019). https://doi.org/10.1109/JSYST.2018.2794462
42. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (Feb 1978). https://doi.org/10.1145/357980.358017, https://doi.org/10.1145/357980.358017
43. Rovida, L., Leporati, A.: Transformer-based language models and homomorphic encryption: An intersection with bert-tiny. In: IWSPA '24: Proceedings of the 10th ACM International Workshop on Security and Privacy Analytics. pp. 3–13 (2024). https://doi.org/10.1145/3643651.3659893
44. Rudman, J.: The state of authorship attribution studies: Some problems and solutions. Computers and the Humanities **31**(4), 351–365 (1998), https://www.jstor.org/stable/30200436
45. Savoy, J.: Machine Learning Methods for Stylometry: Authorship Attribution and Author Profiling. Springer Cham, 1 edn. (2020). https://doi.org/10.1007/978-3-030-53360-1
46. Schlüter, B., Sridhara, S., Kuhne, M., Bertschi, A., Shinde, S.: Heckler: Breaking confidential vms with malicious interrupts. In: 33rd USENIX Security Symposium (USENIX Security '24). pp. 3459–3476. USENIX Association (2024), https://www.usenix.org/system/files/usenixsecurity24-schluter.pdf

47. Sridhara, S., Bertschi, A., Schlüter, B., Shinde, S.: Sigy: Breaking intel sgx enclaves with malicious exceptions and signals. In: ASIA CCS '25: Proceedings of the 20th ACM Asia Conference on Computer and Communications Security. pp. 1643–1658. Association for Computing Machinery (2025). https://doi.org/10.1145/3708821.3710838
48. Stamatatos, E.: A survey of modern authorship attribution methods. Journal of the American Society for Information Science and Technology **60**(3), 538–556 (2008). https://doi.org/10.1002/asi.21001
49. Wosah, P.N., Mirza, Q.A., Sayers, W.: Analysing the email data using stylometric method and deep learning to mitigate phishing attack. International Journal of Information Technology **17**(7), 3823–3834 (2025). https://doi.org/10.1007/s41870-024-01839-5