Privacy-Preserving Classification of Personal Text Messages with Secure Multi-Party Computation: An Application to Hate-Speech Detection

Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, Anderson C. A. Nascimento

***Abstract*—Classification of personal text messages has many useful applications in surveillance, e-commerce, and mental health care, to name a few. Giving applications access to personal texts can easily lead to (un)intentional privacy violations. We propose the first privacy-preserving solution for text classification that is provably secure. Our method, which is based on Secure Multiparty Computation (SMC), encompasses both feature extraction from texts, and subsequent classification with logistic regression and tree ensembles. We prove that when using our secure text classification method, the application does not learn anything about the text, and the author of the text does not learn anything about the text classification model used by the application beyond what is given by the classification result itself. We perform end-to-end experiments with an application for detecting hate speech against women and immigrants, demonstrating excellent runtime results without loss of accuracy.**

***Index Terms*—Text classification, privacy-preserving, secure multiparty computation.**

I. Introduction

The ability to elicit information through automated scanning of personal texts has significant economic and societal value. Machine learning (ML) models for classification of text such as e-mails and SMS messages can be used to infer whether the author is depressed [48], suicidal [44], a terrorist threat [2], or whether the e-mail is a spam message [3], [52]. Other valuable applications of text message classification include user profiling for tailored advertising [34], detection of hate speech [7], and detection of cyberbullying [54]. Some of the above are integrated in parental control applications[1] that monitor text messages on the phones of children and alert their parents when content related to drug use, sexting, suicide etc. is detected. Regardless of the clear benefits, giving applications access to one's personal text messages and e-mails can easily lead to (un)intentional privacy violations.

In this paper, we propose the first privacy-preserving (PP) solution for text classification that is provably secure. To the best of our knowledge, there are no existing Differential Privacy (DP) or Secure Multiparty Computation (SMC) based solutions for PP feature extraction and classification of unstructured texts; the only existing method is based on Homomorphic Encryption (HE) and takes 19 minutes to classify a tweet [17] while leaking information about the text being classified. In our SMC based solution, there are two parties, nick-named *Alice* and *Bob* (see Fig. 1). Bob has a trained ML model that can automatically classify texts. Our secure text classification protocol allows to classify a personal text written by Alice with Bob's ML model in such a way that Bob does not learn anything about Alice's text and Alice does not learn anything about Bob's model. Our solution relies on PP protocols for feature extraction from text and PP machine learning model scoring, which we propose in this paper.

We perform end-to-end experiments with an application for PP detection of hate speech against women and immigrants in text messages. In this use case, Bob has a trained logistic regression (LR) or AdaBoost model that flags hateful texts based on the occurrence of particular words. LR models on word n-grams have been observed to perform comparably to more complex CNN and LSTM model architectures for hate speech detection [37]. Using our protocols, Bob can label Alice's texts as hateful or not without learning which words occur in Alice's texts, and Alice does not learn which words are in Bob's hate speech lexicon, nor how these words are used in the classification process. Moreover, classification is done in seconds, which is two orders of magnitude better than the existing HE solution despite the fact we use over 20 times more features and do not leak any information about Alice's text to the model owner (Bob). The solution based on HE leaks which words in the text are present in Bob's lexicon [17].

We build our protocols using a privacy-preserving machine learning (PPML) framework based on SMC developed by us.[2] All the existing building blocks can be composed within themselves or with new protocols added to the framework. On top of existing building blocks, we also propose a novel protocol for binary classification over binary input features with an ensemble of decisions stumps. While some of our building blocks have been previously proposed, the main contribution of this work consists of the careful choice of the ML techniques, feature engineering and algorithmic and implementation optimizations to enable end-to-end practical PP text classification . Additionally, we provide security definitions and proofs for our proposed protocols.

A conference version of this work appeared at NeurIPS 2019 [49]. This full version contains the security analysis as well

Devin Reich, Ariel Todoki, Martine De Cock, Anderson C. A. Nascimento are with the School of Engineering and Technology, University of Washington Tacoma, Tacoma, WA 98402. Emails: {dreich,atodoki,mdecock,andclay}@uw.edu

Rafael Dowsley is with the Faculty of Information Technology, Monash University, Melbourne, Australia. Email: rafael.dowsley@monash.edu

Martine De Cock is a Guest Professor at Dept. of Applied Mathematics, Computer Science, and Statistics, Ghent University

[1]https://www.bark.us/, https://kidbridge.com/, https://www.webwatcher.com/

[2]https://bitbucket.org/uwtppml

as a new more efficient implementation in Rust.

II. Related work

The interest in privacy-preserving machine learning (PPML) has grown substantially over the last decade. The best-known results in PPML are based on differential privacy (DP), a technique that relies on adding noise to answers, to prevent an adversary from learning information about any particular individual in the dataset from revealed aggregate statistics [32]. While DP in an ML setting aims at protecting the privacy of individuals in the training dataset, our focus is on protecting the privacy of new user data that is classified with proprietary ML models. To this end, we use Secure Multiparty Computation (SMC) [18], a technique in cryptography that has successfully been applied to various ML tasks with structured data (see e.g. [15], [21], [23], [42] and references therein).

To the best of our knowledge there are no existing DP or SMC based solutions for PP feature extraction and classification of unstructured texts. Defenses against authorship attribution attacks that fulfill DP in text classification have been proposed [56]. These methods rely on distortion of term frequency vectors and result in loss of accuracy. In this paper we address a different challenge: we assume that Bob knows Alice, so no authorship obfuscation is needed. Instead, we want to process Alice's text with Bob's classifier, without Bob learning what Alice wrote, and without accuracy loss. To the best of our knowledge, Costantino et al. [17] were the first to propose PP feature extraction from text. In their solution, which is based on homomorphic encryption (HE), Bob learns which of his lexicon's words are present in Alice's tweets, and classification of a single tweet with a model with less than 20 features takes 19 minutes. Our solution does not leak any information about Alice's words to Bob, and classification is done in seconds, even for a model with 500 features.

Below we present existing work that is related to some of the building blocks we use in our PP text classification protocol (see Section IV-A).

Private equality tests have been proposed in the literature based on several different flavors [4]. They can be based on Yao Gates, Homomorphic Encryption, and generic SMC [55]. In our case, we have chosen a simple protocol that depends solely on additions and multiplications over a binary field. While different (and possibly more efficient) comparison protocols could be used instead, they would either require additional computational assumptions or present a marginal improvement in performance for the parameters used here.

Our private feature extraction can be seen as a particular case of private set intersection (PSI). PSI is the problem of securely computing the intersection of two sets without leaking any information except (possibly) the result, such as identifying the intersection of the set of words in a user's text message with the hate speech lexicon used by the classifier. Several paradigms have been proposed to realize PSI functionality, including a Naive hashing solution, Server-aided PSI, and PSI based on oblivious transfer extension. For a complete survey, we refer to Pinkas et al. [47]. In our protocol for PP text classification, we implement private feature extraction by a straightforward application of our equality test protocol. While more efficient protocols could be obtained by using sophisticated hashing techniques, we have decided to stick with our direct solution since it has no probability of failure and works well for the input sizes needed in our problem. For larger input sizes, a more sophisticated protocol would be a better choice [47].

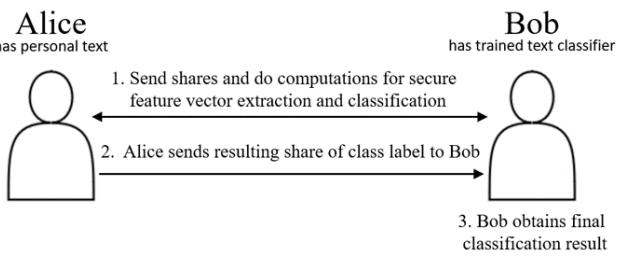

Fig. 1. Roles of Alice and Bob in SMC based text classification

We use two protocols for the secure classification of feature vectors: an existing protocol π_{LR} for *secure classification with LR models* [21]; and a novel *secure AdaBoost classification protocol*. The logistic regression protocol uses solely additions and multiplications over a finite field. The secure AdaBoost classification protocol is an novel optimized protocol that uses solely decision trees of depth one, binary features and a binary output. All these characteristics were used in order to speed up the resulting protocol. The final secure AdaBoost classification protocol uses only two secure inner products and one secure comparison.

Generic protocols for private scoring of machine learning models have been proposed in [9]. The solutions proposed in [9] cannot be used in our setting since they assume that the features' description are publicly known, and thus can be computed locally by Alice and Bob. However, in our case, the features themselves are part of the model and cannot be made public.

Finally, we note that while we implemented our protocols using our own framework for privacy-preserving machine learning[3], any other generic framework for SMC could be also used in principle [50], [24], [43].

III. Preliminaries

We consider *honest-but-curious adversaries*, as is common in SMC based PPML (see e.g. [21], [23]). An honest-but-curious adversary follows the instructions of the protocol, but tries to gather additional information. Secure protocols prevent the latter.

We perform SMC using additively secret shares to do computations modulo an integer q. A value x is secret shared over $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ between parties Alice and Bob by picking $x_A, x_B \in \mathbb{Z}_q$ uniformly at random subject to the constraint that $x = x_A + x_B \mod q$, and then revealing x_A to Alice and x_B to Bob. We denote this secret sharing by $[\![x]\!]_q$, which can be thought of as a shorthand for (x_A, x_B). Secret-sharing based SMC works by first having the parties split their respective *inputs* in secret shares and send some of

[3]https://bitbucket.org/uwtppml

these shares to each other. Naturally, these inputs have to be mapped appropriately to $\mathbb{Z}_q$. Next, Alice and Bob represent the *function* they want to compute securely as a circuit consisting of addition and multiplication gates. Alice and Bob will perform secure additions and multiplications, gate by gate, over the shares until the desired outcome is obtained. The final result can be recovered by combining the final shares, and disclosed as intended, i.e. to one of the parties or to both. It is also possible to keep the final result distributed over shares.

In SMC based text classification, as illustrated in Fig. 1, Alice's *input* is a personal text x and Bob's *input* is an ML model $\mathcal{M}$ for text classification. The function that they want to compute securely is $f(x, \mathcal{M}) = \mathcal{M}(x)$, i.e. the class label of x when classified by $\mathcal{M}$. To this end, Alice splits the text in secret shares while Bob splits the ML model in secret shares. Both parties engage in a protocol in which they send some of the input shares to each other, do local computations on the shares, and repeat this process in an iterative fashion over shares of intermediate results (Step 1). At the end of the joint computations, Alice sends her share of the computed class label to Bob (Step 2), who combines it with his share to learn the classification result (Step 3). As mentioned above, the protocol for Step 1 involves representing the function f as a circuit of addition and multiplication gates.

Given two secret sharings $[\![x]\!]_q$ and $[\![y]\!]_q$, Alice and Bob can locally compute in a straightforward way a secret sharing $[\![z]\!]_q$ corresponding to $z = x + y$ or $z = x - y$ by simply adding/subtracting their local shares of x and y modulo q. Given a constant c, they can also easily locally compute a secret sharing $[\![z]\!]_q$ corresponding to $z = cx$ or $z = x + c$: in the former case Alice and Bob just multiply their local shares of x by c; in the latter case Alice adds c to her share of x while Bob keeps his original share. These local operations will be denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q + [\![y]\!]_q$, $[\![z]\!]_q \leftarrow [\![x]\!]_q - [\![y]\!]_q$, $[\![z]\!]_q \leftarrow c[\![x]\!]_q$ and $[\![z]\!]_q \leftarrow [\![x]\!]_q + c$, respectively. To allow for very efficient secure multiplication of values via operations on their secret shares (denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q[\![y]\!]_q$), we use a trusted initializer that pre-distributes correlated randomness to the parties participating in the protocol before the start of Step 1 in Fig. 1. The initializer is not involved in any other part of the execution and does not learn any data from the parties. This can be straightforwardly extended to efficiently perform secure multiplication of secret shared matrices. The protocol for secure multiplication of secret shared matrices is denoted by π_{DMM} and for the special case of inner-product computation by π_{IP}. Details about the (matrix) multiplication protocol can be found in [21]. We note that if a trusted initializer is not available or desired, Alice and Bob can engage in pre-computations to securely emulate the role of the trusted initializer, at the cost of introducing computational assumptions in the protocol [21].

IV. Secure text classification

Our general protocol for PP text classification relies on several building blocks that are used together to accomplish Step 1 in Fig. 1: a secure equality test, a secure comparison test, private feature extraction, secure protocols for converting between secret sharing modulo 2 and modulo $q > 2$, and private classification protocols. Several of these building blocks have been proposed in the past. However, to the best of our knowledge, this is the very first time they are combined in order to achieve efficient text classification with provable security.

We assume that Alice has a personal text message, and that Bob has a LR or AdaBoost classifier that is trained on unigrams and bigrams as features. Alice constructs the set $A = \{a_1, a_2, \ldots, a_m\}$ of unigrams and bigrams occurring in her message, and Bob constructs the set $B = \{b_1, b_2, \ldots, b_n\}$ of unigrams and bigrams that occur as features in his ML model. We assume that all a_j and b_i are in the form of bit strings. To achieve this, Alice and Bob convert each unigram and bigram on their end to a number N using SHA 224 [46], strictly for its ability to map the same inputs to the same outputs in a pseudo-random manner. Next Alice and Bob map each N on their end to a number between 0 and $2^l - 1$, i.e. a bit string of length l, using a random function in the universal hash family proposed by Carter and Wegman [13].[4] In the remainder we use the term "word" to refer to a unigram or bigram, and we refer to the set $B = \{b_1, b_2, \ldots, b_n\}$ as Bob's lexicon.

Below we outline the protocols for PP text classification. A correctness and security analysis of the protocols is provided in the next section. In the description of the protocols in this paper, we assume that Bob needs to learn the result of the classification, i.e. the class label, at the end of the computations. It is important to note that the protocols described below can be straightforwardly adjusted to a scenario where Alice instead of Bob has to learn the class label, or even to a scenario where neither Alice nor Bob should learn what the class label is and instead it should be revealed to a third party or kept in a secret sharing form. All these scenarios might be relevant use cases of PP text classification, depending on the specific application at hand.

A. Cryptographic building blocks

a) Secure Equality Test: At the start of the secure equality test protocol, Alice and Bob have secret shares of two bit strings $x = x_\ell \ldots x_1$ and $y = y_\ell \ldots y_1$ of length ℓ. x corresponds to a word from Alice's message and y corresponds to a feature from Bob's model. The bit strings x and y are secret shared over $\mathbb{Z}_2$. Alice and Bob follow the protocol to determine whether $x = y$. The protocol π_{EQ} outputs a secret sharing of 1 if $x = y$ and of 0 otherwise.

Protocol π_{EQ}

- For $i = 1, \ldots, \ell$, Alice and Bob locally compute $[\![r_i]\!]_2 \leftarrow [\![x_i]\!]_2 + [\![y_i]\!]_2 + 1$.
- Alice and Bob use secure multiplication to compute a secret sharing of $z = r_1 \cdot r_2 \cdot \ldots \cdot r_\ell$. If $x = y$, then $r_i = 1$ for all bit positions i, hence $z = 1$; otherwise some $r_i = 0$ and therefore $z = 0$. The result is the secret sharing $[\![z]\!]_2$, which is the desired output of the protocol.

[4]The hash function is defined as $((a \cdot N + b) \mod p) \mod 2^l - 1$ where p is a prime and a and b are random numbers less than p. In our experiments, $p = 1,301,081$, $a = 972$, and $b = 52,097$.

This protocol for equality test is folklore in the field of SMC. The $\ell - 1$ multiplications can be organized in as binary tree with the result of the multiplication at the root of the tree. In this way, the presented protocol has $\log(\ell)$ rounds. While there are equality test protocols that have a constant number of rounds, the constant is prohibitively large for the parameters used in our implementation.

b) Secure Feature Vector Extraction: At the start of the feature extraction protocol π_{FE}, Alice has a set $A = \{a_1, a_2, \ldots, a_m\}$ and Bob has a set $B = \{b_1, b_2, \ldots, b_n\}$. A is a set of bit strings that represent Alice's text, and B is a set of bit strings that represent Bob's lexicon. Bob would like to extract words from Alice's text that appear in his lexicon. At the end of the protocol, Alice and Bob have secret shares of a binary feature vector x which represents what words in Bob's lexicon appear in Alice's text. The binary feature vector x of length n is defined as

$$x_i = \begin{cases} 1 & \text{if } b_i \in A \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Protocol π_{FE}

- Alice and Bob secret share each a_j $(j = 1, \ldots, m)$ and each b_i $(i = 1, \ldots, n)$ with each other.
- For $i = 1 \ldots n$:
  - For $j = 1 \ldots m$, Alice and Bob run the secure equality test protocol π_{EQ} to compute secret shares $x_{ij} = 1$ if $a_j = b_i$; $x_{ij} = 0$ otherwise.
  - Alice and Bob locally compute the secret share $[\![x_i]\!]_2 \leftarrow \sum_{j=1}^{m} [\![x_{ij}]\!]_2$.

The secure feature vector extraction can be seen as a private set intersection where the intersection is not revealed but shared [14], [33]. Our solution π_{FE} is tailored to be used within our PPML framework (it uses only binary operations, it is secret sharing based, and is based on pre-distributed binary multiplications). In principle, other protocols could be used here. The efficiency of our protocol can be improved by using hashing techniques [47] at the cost of introducing a small probability of error. The improvements due to hashing are asymptotic and for the parameters used in our fastest running protocol these improvements were not noticeable. Thus, we restricted ourselves to the original protocol without hashing and without any probability of failure.

c) Secure Comparison Test: In our privacy-preserving AdaBoost classifier we will use a secure comparison protocol as a building block. At the start of the secure comparison test protocol, Alice and Bob have secret shares over $\mathbb{Z}_2$ of two bit strings $x = x_\ell \ldots x_1$ and $y = y_\ell \ldots y_1$ of length ℓ. They run the secure comparison protocol π_{DC} of Garay et al. [36] with secret sharings over $\mathbb{Z}_2$ and obtain a secret sharing of 1 if $x \geq y$ and of 0 otherwise.

d) Secure Conversion between $\mathbb{Z}_q$ and $\mathbb{Z}_2$: Some of our building blocks perform computations using secret shares over $\mathbb{Z}_2$ (secure equality test, comparison and feature extraction), while the secure inner product works over $\mathbb{Z}_q$ for $q > 2$. In order to be able to integrate these building blocks we need: (1) A secure bit-decomposition protocol for secure conversion from $\mathbb{Z}_q$ to $\mathbb{Z}_2$. Alice and Bob have as input a secret sharing $[\![x]\!]_q$ and without learning any information about x they should obtain as output secret sharings $[\![x_i]\!]_2$, where $x_\ell \cdots x_1$ is the binary representation of x. We use the secure bit-decomposition protocol π_{decomp} from De Cock et al. [21]; (2) A protocol for secure conversion from $\mathbb{Z}_2$ to $\mathbb{Z}_q$: Alice and Bob have as an input a secret sharing $[\![x]\!]_2$ of a bit x and need to obtain a secret sharing $[\![x]\!]_q$ of the binary value over a larger field $\mathbb{Z}_q$ without learning any information about x. To this end, we use protocol π_{2toQ}.

Protocol π_{2toQ}

- For the input $[\![x]\!]_2$, let $x_A \in \{0, 1\}$ denote Alice's share and $x_B \in \{0, 1\}$ denote Bob's share.
- Alice creates a secret sharing $[\![x_A]\!]_q$ by picking uniformly random shares that sum to x_A and delivers Bob's share to him, and Bob proceeds similarly to create $[\![x_B]\!]_q$.
- Alice and Bob compute $[\![y]\!]_q \leftarrow [\![x_A]\!]_q [\![x_B]\!]_q$.
- The output is computed as $[\![z]\!]_q \leftarrow [\![x_A]\!]_q + [\![x_B]\!]_q - 2[\![y]\!]_q$.

e) Secure Logistic Regression (LR) Classification: At the start of the secure LR classification protocol, Bob has a trained LR model $\mathcal{M}$ that requires a feature vector x of length n as its input, and produces a label $\mathcal{M}(x)$ as its output. Alice and Bob have secret shares of the feature vector x which represents what words in Bob's lexicon appear in Alice's text. At the end of the protocol, Bob gets the result of the classification $\mathcal{M}(x)$. We use an existing protocol π_{LR} for secure classification with LR models [21].[5]

f) Secure AdaBoost Classification: The setting is the same as above, but the model $\mathcal{M}$ is an AdaBoost ensemble of decision stumps instead of a LR model. While efficient solutions for secure classification with tree ensembles were previously known [35], we can take advantage of specific facts about our use case to obtain a more efficient protocol π_{AB}. In more detail, in our use case: (1) all the decision trees have depth 1 (i.e., decision stumps); (2) each feature x_i is binary and therefore when it is used in a decision node, the left and right children correspond exactly to $x_i = 0$ and $x_i = 1$; (3) the output class is binary; (4) the feature values were extracted in a PP way and are secret shared so that no party alone knows their values. We can use the above facts in order to perform the AdaBoost classification by computing two inner products and then comparing their values.

[5] In our case the result of the classification is disclosed to Bob (the party that owns the model) instead of Alice (who has the original input to be classified) as in [21], however it is trivial to modify their protocol so that the final secret share is open towards Bob instead of Alice. Note also that in our case, the feature vector that is used for the classification is already secret shared between Alice and Bob, while in their protocol Alice holds the feature vector, which is then secret shared in the first step of the protocol. This modification is also trivial and does not affect the security of the protocol.

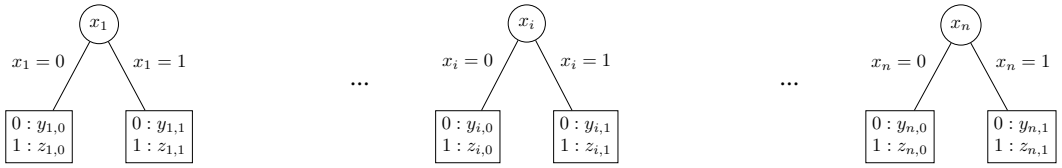

Fig. 2. Ensemble of decision stumps. Each root corresponds to a feature x_i. The leaves contain weights $y_{i,k}$ for the votes for class label 0 and weights $z_{i,k}$ for the votes for class label 1.

Protocol π_{AB}

- Alice and Bob hold secret sharings $[\![x_i]\!]_q$ of each of the n binary features x_i. Bob holds the trained AdaBoost model which consists of two weighted probability vectors $y = (y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$ and $z = (z_{1,0}, z_{1,1}, \ldots, z_{n,0}, z_{n,1})$. For the i-th decision stump: $y_{i,k}$ is the weighted probability (i.e., a probability multiplied by the weight of the i-th decision stump) that the model assigns to the output class being 0 if $x_i = k$, and $z_{i,k}$ is defined similarly for the output class 1 (see Fig. 2).
- Bob secret shares the elements of y and z, and Alice and Bob locally compute secret sharings $[\![w]\!]_q$ of the vector $w = (1 - x_1, x_1, 1 - x_2, x_2, \ldots, 1 - x_n, x_n)$.
- Using the secure inner product protocol π_{IP}, Alice and Bob compute secret sharings of the inner product p_0 between y and w, and of the inner product p_1 between z and w. p_0 and p_1 are the aggregated votes for class label 0 and 1 respectively.
- Alice and Bob use π_{decomp} to compute bitwise secret sharings of p_0 and p_1 over $\mathbb{Z}_2$.
- Alice and Bob use π_{DC} to compare p_1 and p_0, getting as output a secret sharing of the output class c, which is then open towards Bob.

To the best of our knowledge, this is the most efficient provably secure protocol for binary classification over binary input features with an ensemble of decisions stumps.

B. Privacy-preserving classification of personal text messages

We now present our novel protocols for PP text classification. They result from combining the cryptographic building blocks we introduced previously. The PP protocol $\pi_{\mathsf{TC-LR}}$ for classifying the text using a logistic regression model works as follows:

Protocol $\pi_{\mathsf{TC-LR}}$

- Alice and Bob execute the secure feature extraction protocol π_{FE} with input sets A and B in order to obtain secret shares $[\![x_i]\!]_2$ of the feature vector x.
- They run the protocol π_{2toQ} to obtain shares $[\![x_i]\!]_q$ over $\mathbb{Z}_q$.
- Alice and Bob run the secure logistic regression classification protocol π_{LR} in order to get the result of the classification. The LR model $\mathcal{M}$ is given as input to π_{LR} by Bob, and the secret shared feature vector x by both of them. Bob gets the result of the classification $\mathcal{M}(x)$.

The privacy-preserving protocol $\pi_{\mathsf{TC-AB}}$ for classifying the text using AdaBoost works as follows:

Protocol $\pi_{\mathsf{TC-AB}}$

- Alice and Bob execute the secure feature extraction protocol π_{FE} with input sets A and B in order to obtain the secret shares $[\![x_i]\!]_2$ of the feature vector x.
- They run the protocol π_{2toQ} to obtain shares $[\![x_i]\!]_q$ over $\mathbb{Z}_q$.
- Alice and Bob run the secure AdaBoost classification protocol π_{AB} to obtain the result of the classification. The secret shared feature vector x is given as input to π_{AB} by both of them, and the two weighted probability vectors $y = (y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$ and $z = (z_{1,0}, z_{1,1}, \ldots, z_{n,0}, z_{n,1})$ that constitute the model are specified by Bob. Bob gets the output class c.

V. Correctness and Security Analysis of Protocols

A. Security Model

The gold standard model for proving the security of cryptographic protocols nowadays is the Universal Composability (UC) framework [10] and it is the security model that we use in this work. Protocols that are proven UC-secure enjoy strong securities guarantees and can be arbitrary composed without compromising the security. In short, it is the most adequate model to use when the protocols need to be executed in complex environments such as the Internet, and it additionally allows a modular design of bigger protocols. In this work protocols with two parties, Alice and Bob, are considered and in the following we present an overview of the UC framework for this setting. We refer interested readers to the book of Cramer et al. [18] for more details and the most general definitions.

Apart from the protocol participants, Alice and Bob, there are also an adversary $\mathcal{A}$, an ideal world adversary $\mathcal{S}$ (also known as the simulator) and an environment $\mathcal{Z}$ (which captures everything that happens outside of the instance of the protocol that is being analyzed, and therefore is the one giving the inputs and getting the outputs from the protocol). All these entities are assumed to be interactive Turing machines. The network is assumed to be under adversarial control and therefore $\mathcal{A}$ is the one that delivers the messages between Alice and Bob.

In addition to controlling the network scheduling, $\mathcal{A}$ can also corrupt Alice or Bob, in which case he gains the total control over the corrupted party and learn its complete state. For defining the security of the protocol, an ideal functionality $\mathcal{F}$ is defined, which captures the idealized version of what the protocol is supposed to achieve and communicates directly with Alice and Bob to receive the inputs and delivering the outputs of the protocol (in the ideal world, that is all that Alice and Bob do). Then to prove the security of the protocol π, we show that for every possible adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between a real world execution with Alice, Bob and the adversary $\mathcal{A}$ running the protocol π and the ideal world execution with the ideal functionality $\mathcal{F}$, the simulator $\mathcal{S}$ and the dummy version of Alice and Bob that just forward the inputs and outputs between $\mathcal{F}$ and $\mathcal{S}$. Formally:

Definition 5.1 ([10]): A protocol π UC-realizes an ideal functionality $\mathcal{F}$ if, for every possible adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that, for every possible environment $\mathcal{Z}$, the view of the environment $\mathcal{Z}$ in the real world execution with $\mathcal{A}$, Alice and Bob executing the protocol π (with security parameter λ) is computationally indistinguishable from the view of $\mathcal{Z}$ in the ideal world execution with the functionality $\mathcal{F}$, the simulator $\mathcal{S}$ and the dummy Alice and Bob, where the probability distribution is taken over the randomness used by all entities.

Adversarial Model: We consider honest-but-curious adversaries. Honest-but-curious adversaries follow the protocol instructions correctly, but try to learn additional information. We only consider static adversaries, for which the set of corrupted parties is chosen before the start of the protocol execution and does not change. A version of the UC theorem for the case of honest-but-curious adversaries is given in Theorem 4.20 of Cramer et al. [18].

Setup Assumption: It is a well-known fact that secure two-party computation (and also secure multi-party computation) can only achieve UC-security using a setup assumption [11], [12]. Multiple setup assumptions were used previously to achieve UC-security for secure computation protocols, including: the availability of a common reference string [11], [12], [45], the availability of a public-key infrastructure [5], the random oracle model [38], [6], the existence of noisy channels between the parties [27], [31], and the availability of signature cards [39] or tamper-proof hardware [41], [25], [28]. In this work the commodity-based model [8] is used as the setup assumption. [6] In this model there exists a trusted initializer that pre-distributed correlated randomness to Alice and Bob during a setup phase. This setup phase is run before the protocol execution (and in fact can be performed even before Alice and Bob get to know their inputs), and the trusted initializer does not participate in any other point of the protocol. The trusted initializer is modeled by the ideal functionality $\mathcal{F}_{\mathsf{TI}}^{\mathcal{D}}$.

[6]The commodity-based model was used in many other works, e.g., [51], [30], [29], [40], [53], [22], [19], [20], [35], [21], [1], [16].

Functionality $\mathcal{F}_{\mathsf{TI}}^{\mathcal{D}}$

$\mathcal{F}_{\mathsf{TI}}^{\mathcal{D}}$ is parametrized by an algorithm $\mathcal{D}$. Upon initialization run $(D_A, D_B) \xleftarrow{\$} \mathcal{D}$ and deliver D_A to Alice and D_B to Bob.

Simplifications: The simulation strategy in our proofs is in fact very simple: all the computations are performed using secret sharings and all the protocol messages look uniformly random from the point of view of the receiver, with the single exception of the openings of the secret sharings. Nevertheless, the messages that open a secret sharing can be straightforwardly simulated using the outputs of the respective functionalities. In the ideal world, the simulator $\mathcal{S}$ has the leverage of being the one responsible for simulating all the ideal functionalities other than the one whose security is being analyzed (including the trusted initializer functionality $\mathcal{F}_{\mathsf{TI}}^{\mathcal{D}}$), and he can easily use this fact to perform a perfect simulation. For this reason the real and ideal world are indistinguishable for any environment $\mathcal{Z}$ and achieve perfect security.

The messages of the ideal functionalities are formally public delayed outputs, i.e., first the simulator is asked whether it allows the message to be delivered (this is due to the fact that in the real world the adversary controls the scheduling of the network), and the message is only delivered when $\mathcal{S}$ agrees. And formally, every instance has a session identification. We omit those information from descriptions for the sake of readability.

Security of the Building Blocks: The protocol for secure distributed matrix multiplication π_{DMM} UC-realizes the distributed matrix multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ [26], [21].

Functionality $\mathcal{F}_{\mathsf{DMM}}$

$\mathcal{F}_{\mathsf{DMM}}$ is executed with Alice and Bob is parametrized by the size q of the ring and the dimensions (i, j) and (j, k) of the matrices.

Input: Upon receiving a message from Alice/Bob with her/his shares of $[\![X]\!]_q$ and $[\![Y]\!]_q$, verify if the share of X is in $\mathbb{Z}_q^{i\times j}$ and the share of Y is in $\mathbb{Z}_q^{j\times k}$. If it is not, abort. Otherwise, record the shares, ignore any subsequent message from that party and inform the other party about the receipt.

Output: Upon receipt of the inputs from both Alice and Bob, reconstruct X and Y from the shares, compute $Z = XY$ and create a secret sharing $[\![Z]\!]_q$. Before the deliver of the output shares, a corrupt party fix its share of the output to any constant value. In both cases the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

The protocol for secure comparison π_{DC} UC-realizes the functionality $\mathcal{F}_{\mathsf{DC}}$ [36], [21].

Functionality $\mathcal{F}_{\mathsf{DC}}$

$\mathcal{F}_{\mathsf{DC}}$ is parametrized by the bit-length ℓ of the values being compared.

Input: Upon receiving a message from Alice/Bob with her/his shares of $[\![x_i]\!]_2$ and $[\![y_i]\!]_2$ for all $i \in \{1, \ldots, \ell\}$, record the shares, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon receipt of the inputs from both parties, reconstruct x and y from the bitwise shares. If $x \geq y$, then create and distribute to Alice and Bob the secret sharing $[\![1]\!]_2$; otherwise the secret sharing $[\![0]\!]_2$. Before the deliver of the output shares, a corrupt party fix its share of the output to any constant value. In both cases the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

The protocol for secure bit-decomposition π_{decomp} UC-realizes the functionality $\mathcal{F}_{\mathsf{decomp}}$ [21].

Functionality $\mathcal{F}_{\mathsf{decomp}}$

$\mathcal{F}_{\mathsf{decomp}}$ is parametrized by the bit-length ℓ of the value x being converted from an additive secret sharing $[\![x]\!]_q$ in $\mathbb{Z}_q$ to additive bitwise secret sharings $[\![x_i]\!]_2$ in $\mathbb{Z}_2$ such that $x = x_\ell \cdots x_1$.

Input: Upon receiving a message from Alice or Bob with her/his share of $[\![x]\!]_q$, record the share, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon receipt of both shares, reconstruct x, compute its bitwise representation $x_\ell \cdots x_1$, and for $i \in \{1, \ldots, \ell\}$ distribute new secret sharings $[\![x_i]\!]_2$ of the bit x_i. Before the output deliver, the corrupt party fix its shares of the outputs to any constant values. The shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraints.

The LR classification protocol π_{LR} UC-realizes the functionality $\mathcal{F}_{\mathsf{LR}}$ [21].

Functionality $\mathcal{F}_{\mathsf{LR}}$

$\mathcal{F}_{\mathsf{LR}}$ computes the classification according to a logistic regression model with the threshold value set to 0.5. The input feature vector x is secret shared between Alice and Bob.

Input: Upon receiving the weight vector w, the intercept value b and his shares $[\![x_i]\!]_q$ of the elements of x from Bob, or her shares $[\![x_i]\!]_q$ of the elements of x from Alice, store the information, ignore any subsequent message from that party, and inform the other party about the receipt.

Output: Upon getting the inputs from both parties, reconstruct the feature vector x, compute the value $\mathsf{sign}\left(\langle x, w\rangle + b\right)$ and output it to Bob as the class prediction.

We now show that the equality test protocol π_{EQ} UC-realizes functionality $\mathcal{F}_{\mathsf{EQ}}$.

Functionality $\mathcal{F}_{\mathsf{EQ}}$

$\mathcal{F}_{\mathsf{EQ}}$ is parametrized by the bit-length ℓ of the values being compared.

Input: Upon receiving a message from Alice/Bob with her/his shares of $[\![x_i]\!]_2$ and $[\![y_i]\!]_2$ for all $i \in \{1, \ldots, \ell\}$, record the shares, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon receipt of the inputs from both parties, reconstruct x and y from the bitwise shares. If $x = y$, then create and distribute to Alice and Bob the secret sharing $[\![1]\!]_2$; otherwise the secret sharing $[\![0]\!]_2$. Before the deliver of the output shares, a corrupt party fix its share of the output to any constant value. In both cases the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

The correctness of the equality test protocol π_{EQ} follows from the fact that in the case that $x = y$, then all r_i's will be equal to 1 and therefore $z = \prod_i r_i$ will also be 1. If $x \neq y$, then for at least one value i, we have that $r_i = 0$, and therefore $z = 0$. For the simulation, $\mathcal{S}$ executes an internal copy of $\mathcal{A}$ interacting with an instance of π_{EQ} in which the uncorrupted parties use dummy inputs. Note that all the messages that $\mathcal{A}$ receives look uniformly random to him. Since the share multiplication protocol is substituted by $\mathcal{F}_{\mathsf{DMM}}$ using the UC composition theorem, and $\mathcal{S}$ is the one responsible for simulating $\mathcal{F}_{\mathsf{DMM}}$ in the ideal world, $\mathcal{S}$ can leverage this fact in order to extract the share that any corrupted party have of the value $x_i + y_i$, let the extracted value of the corrupted party be denoted by $v_{i,C}$. The simulator then pick random values $x_{i,C}, y_{i,C} \in \{0, 1\}$ such that $x_{i,C} + y_{i,C} = v_{i,C} \mod 2$ and submit these values to $\mathcal{F}_{\mathsf{EQ}}$ as being the shares of the corrupted party for x_i and y_i (note that the result of $\mathcal{F}_{\mathsf{EQ}}$ only depends on the values of $x_i + y_i \mod 2$). $\mathcal{S}$ is also able to fix the output share of the corrupted party in $\mathcal{F}_{\mathsf{EQ}}$ so that it matches the one in the instance of π_{EQ}. This is a perfect simulation strategy, no environment $\mathcal{Z}$ can distinguish the ideal and real worlds and therefore π_{EQ} UC-realizes $\mathcal{F}_{\mathsf{EQ}}$.

Next, we prove that the secure feature extraction protocol π_{FE} UC-realizes functionality $\mathcal{F}_{\mathsf{FE}}$.

Functionality $\mathcal{F}_{\mathsf{FE}}$

$\mathcal{F}_{\mathsf{FE}}$ is parametrized by the sizes m of Alice's set and n of Bob's set, and the bit-length ℓ of the elements.

Input: Upon receiving a message from Alice with her set $A = \{a_1, a_2, \ldots, a_m\}$ or from Bob with his set $B = \{b_1, b_2, \ldots, b_n\}$, record the set, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon receipt of the inputs from both parties, define the binary feature vector x of length n by setting each element x_i to 1 if $b_i \in A$, and to 0 otherwise. Then create and distribute to Alice and Bob the secret sharings $[\![x_i]\!]_2$. Before the deliver of the output shares, a corrupt party fix its share of the output to any constant value. In both cases the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

The correctness of the secure feature extraction protocol π_{FE} follows directly from the fact that each x_{ij} is equal to 1 if, and only if, $a_j = b_i$, and therefore $x_i = \sum_j x_{ij}$ is equal to 1 if, and only if, b_i is equal to some element of A. In the ideal world, the simulator $\mathcal{S}$ runs internally a copy of $\mathcal{A}$ and an execution of π_{FE} with dummy inputs for the uncorrupted parties. All the messages from the uncorrupted parties look uniformly random from $\mathcal{A}$'s point of view, and therefore the simulation is perfect. $\mathcal{S}$ uses the leverage of being responsible for simulating $\mathcal{F}_{\mathsf{EQ}}$ (π_{EQ} is substituted by $\mathcal{F}_{\mathsf{EQ}}$ using the UC composition theorem) in order to extract the inputs of any corrupted party and forward it to $\mathcal{F}_{\mathsf{FE}}$. No environment $\mathcal{Z}$ can distinguish the ideal world from the real one, and thus π_{FE} UC-realizes $\mathcal{F}_{\mathsf{FE}}$.

The conversion protocol π_{2toQ} UC-realizes functionality $\mathcal{F}_{\mathsf{2toQ}}$.

Functionality $\mathcal{F}_{\mathsf{2toQ}}$

$\mathcal{F}_{\mathsf{2toQ}}$ is parametrized by the size of the field q.

Input: Upon receiving a message from Alice/Bob with her/his share of $[\![x]\!]_2$, record the share, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon receipt of the inputs from both parties, reconstruct x, then create and distribute to Alice and Bob the secret sharing $[\![x]\!]_q$. Before the deliver of the output shares, a corrupt party fix its share of the output to any constant value. In both cases the shares of the uncorrupted parties are then created by picking uniformly random values subject to the correctness constraint.

In the case of the conversion protocol π_{2toQ} the correctness of the protocol execution follows straightforwardly: since $x = x_a + x_B \mod 2$, then $z = x_A + x_B - 2x_Ax_B$ is such that $z = x$ for all possible values $x_A, x_B \in \{0, 1\}$. As for the security, the simulator $\mathcal{S}$ runs internally a copy of the adversary $\mathcal{A}$ and simulates to him an execution of the protocol π_{2toQ} using dummy inputs for the uncorrupted parties. As all the messages from the uncorrupted parties look uniformly random from the adversary point of view, and so the simulation is perfect. The simulator can use the fact that it is the one simulating the multiplication functionality $\mathcal{F}_{\mathsf{DMM}}$ (the secret sharing multiplication is substituted by $\mathcal{F}_{\mathsf{DMM}}$ using the UC composition theorem) in order to extract the share of any corrupted party and fix the input to/output from $\mathcal{F}_{\mathsf{2toQ}}$ appropriately, so that no environment $\mathcal{Z}$ can distinguish the real and ideal worlds. Hence π_{2toQ} UC-realizes $\mathcal{F}_{\mathsf{2toQ}}$.

Finally, the AdaBoost classification protocol π_{AB} UC-realizes functionality $\mathcal{F}_{\mathsf{AB}}$.

Functionality $\mathcal{F}_{\mathsf{AB}}$

$\mathcal{F}_{\mathsf{AB}}$ computes the classification according to AdaBoost with multiple decision stumps. All the features are binary and the output class is also binary. The input feature vector x is secret shared between Alice and Bob. The model specified by Bob can be expressed in a simplified way by two weighted probability vectors $y = (y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$ and $z = (z_{1,0}, z_{1,1}, \ldots, z_{n,0}, z_{n,1})$. For the i-th decision stump: $y_{i,k}$ is the weighted probability (i.e., a probability multiplied by the weight of the i-th decision stump) that the model assigns to the output class being 0 if $x_i = k$, and $z_{i,k}$ is defined similarly for the output class 1.

Input: Upon receiving the vectors y and z and his shares $[\![x_i]\!]_q$ of the elements of the feature vector x from Bob, or her shares $[\![x_i]\!]_q$ of the elements of x from Alice, store the information, ignore any subsequent message from that party, and inform the other party about the receipt.

Output: Upon getting the inputs from both parties, reconstruct the feature vector x and let $w = (1 - x_1, x_1, 1 - x_2, x_2, \ldots, 1 - x_n, x_n)$. If $\langle w, z\rangle \geq \langle w, y\rangle$, output the class prediction 1 to Bob; otherwise output 0.

The AdaBoost classification protocol π_{AB} is trivially correct for the case of binary features and output class, and decision stumps. In the simulation, $\mathcal{S}$ runs an internal copy of $\mathcal{A}$ interacting with a simulated instance of π_{AB} that uses dummy inputs for the uncorrupted parties. π_{IP} is substituted by $\mathcal{F}_{\mathsf{DMM}}$ using the UC composition theorem. $\mathcal{S}$ uses the leverage of simulating $\mathcal{F}_{\mathsf{DMM}}$ in order to extract the shares of the feature vector belonging to a corrupted party, as well as the weighted probability vectors y and z if Bob is corrupted. $\mathcal{S}$ can then give these extracted inputs to $\mathcal{F}_{\mathsf{AB}}$. No environment can distinguish the real and ideal worlds since the simulation is perfect, and thus π_{AB} UC-realizes $\mathcal{F}_{\mathsf{AB}}$.

Security of the Privacy-Preserving Text Classification Solutions: The text classification protocol $\pi_{\mathsf{TC-LR}}$ UC-realizes functionality $\mathcal{F}_{\mathsf{TC-LR}}$.

Functionality $\mathcal{F}_{\mathsf{TC-LR}}$

$\mathcal{F}_{\mathsf{TC-LR}}$ computes the privacy-preserving text classification according to a logistic regression model with the threshold value set to 0.5. It is parametrized by the sizes m of Alice's set and n of Bob's set, and the bit-length ℓ of the elements.

Input: Upon receiving a message from Alice with her set $A = \{a_1, a_2, \ldots, a_m\}$ or from Bob with his set $B = \{b_1, b_2, \ldots, b_n\}$, the weight vector w and the intercept value b, record the values, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon getting the inputs from both parties, define the feature vector x of length n as follows: $x_i = 1$ if $b_i \in A$; and $x_i = 0$ otherwise. Compute the value $\mathsf{sign}\,(\langle x, w\rangle + b)$ and output it to Bob as the class prediction.

The protocol $\pi_{\mathsf{TC-LR}}$ simply executes sequentially the protocols π_{FE}, π_{2toQ} and π_{LR}. Given that these protocols UC-realize $\mathcal{F}_{\mathsf{FE}}$, $\mathcal{F}_{\mathsf{2toQ}}$ and $\mathcal{F}_{\mathsf{LR}}$, respectively, they can be substituted by the functionalities using the UC composition theorem. Note that the sequential composition of those functionalities trivially perform the same computation as $\mathcal{F}_{\mathsf{TC-LR}}$, and no information other than the output of the classification is revealed (all the intermediate values are kept as secret sharings). In the ideal world $\mathcal{S}$ simulates an internal copy of the adversary $\mathcal{A}$ running $\pi_{\mathsf{TC-LR}}$ and using dummy inputs for the uncorrupted parties. The simulator $\mathcal{S}$ can easily extract all the information (from the corrupted parties) that it needs to provide to $\mathcal{F}_{\mathsf{TC-LR}}$ by using the leverage of being responsible for simulating $\mathcal{F}_{\mathsf{FE}}$, $\mathcal{F}_{\mathsf{2toQ}}$ and $\mathcal{F}_{\mathsf{LR}}$ in the ideal world. Therefore no environment $\mathcal{Z}$ can distinguish the real world from the ideal world, and $\pi_{\mathsf{TC-LR}}$ UC-realizes $\mathcal{F}_{\mathsf{TC-LR}}$.

And the text classification protocol $\pi_{\mathsf{TC-AB}}$ UC-realizes functionality $\mathcal{F}_{\mathsf{TC-AB}}$.

Functionality $\mathcal{F}_{\mathsf{TC-AB}}$

$\mathcal{F}_{\mathsf{TC-AB}}$ computes the privacy-preserving text classification according to AdaBoost with multiple decision stumps. It is parametrized by the sizes m of Alice's set and n of Bob's set, and the bit-length ℓ of the elements. All the features are binary and the output class is also binary. The model specified by Bob can be expressed in a simplified way by two weighted probability vectors $y = (y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$ and $z = (z_{1,0}, z_{1,1}, \ldots, z_{n,0}, z_{n,1})$. For the i-th decision stump: $y_{i,k}$ is the weighted probability (i.e., a probability multiplied by the weight of the i-th decision stump) that the model assigns to the output class being 0 if the feature $x_i = k$, and $z_{i,k}$ is defined similarly for the output class 1.

Input: Upon receiving a message from Alice with her set $A = \{a_1, a_2, \ldots, a_m\}$ or from Bob with his set $B = \{b_1, b_2, \ldots, b_n\}$, y and z, record the values, ignore any subsequent messages from that party and inform the other party about the receipt.

Output: Upon getting the inputs from both parties, define the feature vector x of length n as follows: $x_i = 1$ if $b_i \in A$; and $x_i = 0$ otherwise. Let $w = (1 - x_1, x_1, 1 - x_2, x_2, \ldots, 1 - x_n, x_n)$. If $\langle w, z\rangle \geq \langle w, y\rangle$, output the class prediction 1 to Bob; otherwise output 0.

Similarly to the above case, the protocol $\pi_{\mathsf{TC-AB}}$ just runs sequentially the protocols π_{FE}, π_{2toQ} and π_{AB}, that can be substituted by $\mathcal{F}_{\mathsf{FE}}$, $\mathcal{F}_{\mathsf{2toQ}}$ and $\mathcal{F}_{\mathsf{AB}}$ using the UC composition theorem. The result of the computation is trivially the same as in $\mathcal{F}_{\mathsf{TC-AB}}$, and no additional information is revealed. $\mathcal{S}$ runs internally a copy of $\mathcal{A}$ interacting with a simulated instance of $\pi_{\mathsf{TC-AB}}$ (using dummy inputs for the uncorrupted parties) and can easily extract from the corrupted parties all the information that it must provide to $\mathcal{F}_{\mathsf{TC-AB}}$ by using the leverage of being responsible for simulating $\mathcal{F}_{\mathsf{FE}}$, $\mathcal{F}_{\mathsf{2toQ}}$ and $\mathcal{F}_{\mathsf{AB}}$ in the ideal world. No environment $\mathcal{Z}$ can distinguish the real and ideal worlds, and therefore $\pi_{\mathsf{TC-AB}}$ UC-realizes $\mathcal{F}_{\mathsf{TC-AB}}$.

VI. Experimental results

We evaluate the proposed protocols in a use case for the detection of hate speech in short text messages, using data from [7]. The corpus consists of 10,000 tweets, 60% of which are annotated as hate speech against women or immigrants. We convert all characters to lowercase, and turn each tweet into a set of word unigrams and bigrams. There are 29,853 distinct unigrams and 93,629 distinct bigrams in the dataset, making for a total of 123,482 features.

We implemented the protocols from Section IV in both Java and Rust using the respective versions of Lynx (Java-Lynx and Rust-Lynx).[7] Accuracy results for a variety of models trained to classify a tweet as hate speech vs. non-hate speech are presented in Tables I and II. The models are evaluated using 5-fold cross-validation over the entire corpus of 10,000 tweets. The top rows in Tables I and II correspond to tree ensemble models consisting of 50, 200, and 500 decision stumps respectively; the root of each stump corresponds to a feature. The bottom rows contain results for an LR model trained on 50, 200, and 500 features (preselected based on information gain), and an LR model trained on all features. We ran experiments for feature sets consisting of unigrams and bigrams, as well as for feature sets consisting of unigrams only, observing that the inclusion of bigrams leads to a small improvement in accuracy. Note that designing a model to obtain the highest possible accuracy is

[7] https://bitbucket.org/uwtppml

TABLE I
ACCURACY (ACC) RESULTS USING 5-FOLD CROSS-VALIDATION OVER THE CORPUS OF 10,000 TWEETS. TOTAL TIME (TOT) NEEDED TO SECURELY CLASSIFY A TEXT WITH OUR **JAVA FRAMEWORK**, BROKEN DOWN IN TIME NEEDED FOR FEATURE VECTOR EXTRACTION (EXTR) AND TIME FOR FEATURE VECTOR CLASSIFICATION (CLASS).

Java-Lynx	**Unigrams**				**Unigrams+Bigrams**			
	Acc	**Time (in sec)**			**Acc**	**Time (in sec)**		
		Extr	Class	Tot		Extr	Class	Tot
Ada; 50 trees; depth 1	71.6%	0.8	6.4	7.2	73.3%	1.5	6.6	8.1
Ada; 200 trees; depth 1	73.0%	2.8	6.4	9.2	74.2%	9.4	6.6	16.0
Ada; 500 trees; depth 1	73.9%	6.6	6.7	13.3	**74.4%**	**21.6**	**6.7**	**28.3**
Logistic regression (50 feat.)	**72.4%**	**0.8**	**3.7**	**4.5**	73.8%	1.5	3.8	5.3
Logistic regression (200 feat.)	73.3%	2.8	3.7	6.5	73.7%	9.4	3.8	13.2
Logistic regression (500 feat.)	73.4%	6.6	3.8	10.4	74.2%	21.6	4.1	25.7

TABLE II
ACCURACY (ACC) RESULTS USING 5-FOLD CROSS-VALIDATION OVER THE CORPUS OF 10,000 TWEETS. TOTAL TIME (TOT) NEEDED TO SECURELY CLASSIFY A TEXT WITH OUR **RUST FRAMEWORK**, BROKEN DOWN IN TIME NEEDED FOR FEATURE VECTOR EXTRACTION (EXTR) AND TIME FOR FEATURE VECTOR CLASSIFICATION (CLASS).

Rust-Lynx	**Unigrams**				**Unigrams+Bigrams**			
	Acc	**Time (in sec)**			**Acc**	**Time (in sec)**		
		Extr	Class	Tot		Extr	Class	Tot
Ada; 50 trees; depth 1	71.6%	0.925	0.062	0.987	73.3%	2.583	0.064	2.647
Ada; 200 trees; depth 1	73.0%	3.652	0.062	3.714	74.2%	9.915	0.062	9.977
Ada; 500 trees; depth 1	73.9%	9.227	0.066	9.293	**74.4%**	**25.685**	**0.066**	**25.751**
Logistic regression (50 feat.)	**72.4%**	**0.915**	**0.038**	**0.953**	73.8%	2.583	0.041	2.624
Logistic regression (200 feat.)	73.3%	3.652	0.039	3.691	73.7%	9.915	0.042	9.957
Logistic regression (500 feat.)	73.4%	9.227	0.041	9.268	74.2%	25.685	0.041	25.726

not the focus of this paper. Instead, our goal is to demonstrate that PP text classification based on SMC is feasible in practice.

We ran experiments on AWS c5.9xlarge machines with 36 vCPUs, 72.0 GiB Memory. Each of the parties ran on separate machines (connected with a Gigabit Ethernet network), which means that the results in Table I and II cover communication time in addition to computation time. Each runtime experiment was repeated 3 times and average results are reported. In Table I and II we report the time (in sec) needed for converting a tweet into a feature vector (Extr), for classification of the feature vector (Class), and for the overall process (Tot). Our results showed that for all hyper-parameter choices of the models, our Rust-Lynx implementation outperforms the Java-Lynx implementation, with the exception of the case of logistic regression using unigrams and bigrams with 500 features, where the performance of both were similar.

A. Analysis

The best running times were obtained using unigrams, 50 features and logistic regression (4.5s in Java-Lynx and 0.953s in Rust-Lynx) with an accuracy of 72.4%. The highest accuracy (74.4%) was obtained by using unigram and bigrams, 500 features and AdaBoost with a running time equal to 28.3s in Java-Lynx and 25.751s in Rust-Lynx. From these results, it is clear that feature engineering plays a major role in optimizing privacy-preserving machine learning solutions based on SMC. We managed to reduce the running time from 5,396.8s (logistic regression, unigram and bigrams, all 123,482 features being used) to 2.624s (logistic regression, unigrams and bigrams, 50 features in Rust-Lynx) without any loss in accuracy and to 0.935s (logistic regression, unigrams only, 50 features in Rust-Lynx) with a small loss.

B. Optimizing the computational and communication complexities

The feature extraction protocol requires $n \cdot m$ secure equality tests of bit strings. The equality test relies on secure multiplication, which is the more expensive operation. To reduce the number of required equality tests, Alice and Bob can each first map their bit strings to p buckets $A_1, A_2, \ldots, A_p$ and $B_1, B_2, \ldots, B_p$ respectively, so that bit strings from each A_i need to only be compared with bit strings from B_i. Each bit string a_j and b_i is hashed and the first t bits of the hash output are used to define the bucket number corresponding to that bit string, using a total of $p = 2^t$ buckets. In order not to leak how many elements are mapped to each bucket (which can leak some information about the probability distribution of the elements, as the hash function is known by everyone), each bucket has a fixed number of elements (s_1 for Bob's buckets and s_2 for Alice's buckets) and the empty spots in the buckets are filled up with dummy elements. The feature extraction protocol now requires $p \cdot s_1 \cdot s_2$ equality tests, which can be substantially smaller than $n \cdot m$. When using bucketization, the feature vector of length n from (1) is expanded to a feature vector of length $p \cdot s_1$, containing the original n features as well as the $p \cdot s_1 - n$ dummy features that Bob created to fill up his buckets. These dummy features do not have any effect on the accuracy of the classification because Bob's model does not take them into account: the trees with dummy features in an AdaBoost model have 0 weight for both class labels, and

the dummy features' coefficients in an LR model are always 0.

The size of the buckets has to be chosen sufficiently large to avoid overflow. The choice depends directly on the number $p = 2^t$ of buckets (which is kept constant for Alice and Bob) and the number of elements to be placed in the buckets, i.e. n elements on Bob's side and m elements on Alice's side. While for hash functions coming from a 2-universal family of hash functions the computation of these probabilities is relatively straightforward, the same is not true for more complicated hash functions [47]. In that case, numerical simulations are needed in order to bound the required probability.

The effect of using buckets is more significant for large values of n and m. In our case, after performing feature engineering for reducing the number of elements in each set, in the best case, we end up with inputs for which there is no significant difference between the original protocol (without buckets) and the protocol that uses buckets. If the performance of these two cases is comparable, one is better off using the version without buckets, since there will be no probability of information being leaked due to bucket overflow.

Another way we could possibly improve the communication and computation complexities of the protocol is by reducing the number of bits used to represent each feature albeit at the cost of increasing the probability of collisions (different features being mapped into the same bit strings). We used 13 bits for representing unigrams and 17 bits for representing unigrams and bigrams. We did not observe any collisions.

Finally, we note that if the protocol is to be deployed over a wide area network, rather than a local area network, Yao garbled circuits would become a preferable choice for the round intensive parts of our solution (such as in the private feature extraction part).

VII. Conclusion

In this paper we have presented the first provably secure method for privacy-preserving (PP) classification of unstructured text. We have provided an analysis of the correctness and security of solution. As a side result, we also present a novel protocol for binary classification over binary input features with an ensemble of decisions stumps. An implementation of the protocols in Java, run on AWS machines, allowed us to classify text messages securely within seconds. It is important to note that this run time (1) includes both secure feature extraction and secure classification of the extracted feature vector; (2) includes both computation and communication costs, as the parties involved in the protocol were run on separate machines; (3) is two orders of magnitude better than the only other existing solution, which is based on HE. Our results show that in order to make PP text classification practical, one needs to pay close attention not only to the underlying cryptographic protocols but also to the underlying ML algorithms. ML algorithms that would be a clear choice when used in the clear might not be useful at all when transferred to the SMC domain. One has to optimize these ML algorithms having in mind their use within SMC protocols. Our results provide the first evidence that provably secure PP text classification is feasible in practice.

References

[1] Anisha Agarwal, Rafael Dowsley, Nicholas D. McKinney, Dongrui Wu, Chin-Teng Lin, Martine De Cock, and Anderson C. A. Nascimento. Protecting privacy of users in brain-computer interface applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(8):1546–1555, Aug 2019.

[2] Peter Ray Allison. Tracking terrorists online might invade your privacy. BBC, http://www.bbc.com/future/story/20170808-tracking-terrorists-online-might-invade-your-privacy, 2017.

[3] Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. Contributions to the study of SMS spam filtering: new collection and results. In *Proc. of the 11th ACM Symposium on Document Engineering*, pages 259–262, 2011.

[4] Nuttapong Attrapadung, Goichiro Hanaoka, Shinsaku Kiyomoto, Tomoaki Mimoto, and Jacob CN Schuldt. A taxonomy of secure two-party comparison protocols and efficient constructions. In *15th Annual Conference on Privacy, Security and Trust (PST)*, 2017.

[5] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Annual Symposium on Foundations of Computer Science*, pages 186–195, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[6] Paulo S. L. M. Barreto, Bernardo David, Rafael Dowsley, Kirill Morozov, and Anderson C. A. Nascimento. A framework for efficient adaptively secure composable oblivious transfer in the ROM. Cryptology ePrint Archive, Report 2017/993, 2017. http://eprint.iacr.org/2017/993.

[7] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. Semeval-2019 Task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proc. of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. ACL, 2019.

[8] Donald Beaver. Commodity-based cryptography (extended abstract). In *29th Annual ACM Symposium on Theory of Computing*, pages 446–455, El Paso, TX, USA, May 4–6, 1997. ACM Press.

[9] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[10] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[11] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[12] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[13] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[14] Michele Ciampi and Claudio Orlandi. Combining private set-intersection with secure two-party computation. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 464–482, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.

[15] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.

[16] Martine De Cock, Rafael Dowsley, Anderson C. A. Nascimento, Davis Railsback, Jianwei Shen, and Ariel Todoki. High performance logistic regression for privacy-preserving genome analysis. *IACR Cryptol. ePrint Arch.*, 2020:171, 2020.

[17] Gianpiero Costantino, Antonio La Marra, Fabio Martinelli, Andrea Saracino, and Mina Sheikhalishahi. Privacy-preserving text mining as a service. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 890–897, 2017.

[18] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[19] Bernardo David, Rafael Dowsley, Raj Katti, and Anderson CA Nascimento. Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols. In *International Conference on Provable Security*, pages 354–367. Springer, 2015.

[20] Bernardo David, Rafael Dowsley, Jeroen van de Graaf, Davidson Marques, Anderson C. A. Nascimento, and Adriana C. B. Pinto. Unconditionally secure, universally composable privacy preserving linear algebra. *IEEE Transactions on Information Forensics and Security*, 11(1):59–73, 2016.

[21] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson Nascimento, Wing-Sea Poon, and Stacey Truex. Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. *IEEE Transactions on Dependable and Secure Computing*, 16(2):217–230, 2019.

[22] Martine De Cock, Rafael Dowsley, Anderson C. A. Nascimento, and Stacey C. Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *8th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 3–14, 2015.

[23] Sebastiaan de Hoogh, Berry Schoenmakers, Ping Chen, and Harm op den Akker. Practical secure decision tree learning in a teletreatment application. In *International Conference on Financial Cryptography and Data Security*, pages 179–194. Springer, 2014.

[24] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[25] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 164–181, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany.

[26] Rafael Dowsley. *Cryptography Based on Correlated Data: Foundations and Practice*. PhD thesis, Karlsruhe Institute of Technology, Germany, 2016.

[27] Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. On the possibility of universally composable commitments based on noisy channels. In *SBSEG 2008*, pages 103–114, Gramado, Brazil, September 1–5, 2008.

[28] Rafael Dowsley, Jörn Müller-Quade, and Tobias Nilges. Weakening the isolation assumption of tamper-proof hardware tokens. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15: 8th International Conference on Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 197–213, Lugano, Switzerland, May 2–5, 2015. Springer, Heidelberg, Germany.

[29] Rafael Dowsley, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, Hideki Imai, and Anderson C. A. Nascimento. Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. *IEICE Transactions*, 94-A(2):725–734, 2011.

[30] Rafael Dowsley, Jeroen Van De Graaf, Davidson Marques, and Anderson CA Nascimento. A two-party protocol with trusted initializer for computing the inner product. In *International Workshop on Information Security Applications*, pages 337–350. Springer, 2010.

[31] Rafael Dowsley, Jeroen van de Graaf, Jörn Müller-Quade, and Anderson C. A. Nascimento. On the composability of statistically secure bit commitments. *Journal of Internet Technology*, 14(3):509–516, 2013.

[32] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[33] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. Private set intersection with linear communication from general assumptions. Cryptology ePrint Archive, Report 2018/238, 2018. https://eprint.iacr.org/2018/238.

[34] Golnoosh Farnadi, Geetha Sitaraman, Shanu Sushmita, Fabio Celli, Michal Kosinski, David Stillwell, Sergio Davalos, Marie-Francine Moens, and Martine De Cock. Computational personality recognition in social media. *User Modeling and User-Adapted Interaction*, 26(2-3):109–142, 2016.

[35] Kyle Fritchman, Keerthanaa Saminathan, Rafael Dowsley, Tyler Hughes, Martine De Cock, Anderson Nascimento, and Ankur Teredesai. Privacy-preserving scoring of tree ensembles: A novel framework for AI in healthcare. In *Proc. of 2018 IEEE International Conference on Big Data*, pages 2412–2421, 2018.

[36] Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 330–342, Beijing, China, April 16–20, 2007. Springer, Heidelberg, Germany.

[37] Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. All you need is "love": Evading hate-speech detection. In *Proc. of the 11th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2018.

[38] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 58–76, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[39] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Universally composable zero-knowledge arguments and commitments from signature cards. In *MoraviaCrypt 2005*, 2005.

[40] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Theory of Cryptography*, pages 600–620. Springer, 2013.

[41] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.

[42] Selim V Kaya, Thomas B Pedersen, Erkay Savaş, and Yücel Saygıỳn. Efficient privacy preserving distributed clustering based on secret sharing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 280–291. Springer, 2007.

[43] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[44] Bridianne O'Dea, Stephen Wan, Philip J. Batterham, Alison L. Calear, Cecile Paris, and Helen Christensen. Detecting suicidality on Twitter. *Internet Interventions*, 2(2):183–188, 2015.

[45] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[46] Wouter Penard and Tim van Werkhoven. On the secure hash algorithm family. In *Cryptography in Context*, pages 1–18. 2008.

[47] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):7, 2018.

[48] Andrew G. Reece, Andrew J. Reagan, Katharina L.M. Lix, Peter Sheridan Dodds, Christopher M. Danforth, and Ellen J. Langer. Forecasting the onset and course of mental illness with Twitter data. *Scientific Reports*, 7(1):13006, 2017.

[49] Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, and Anderson C. A. Nascimento. Privacy-preserving classification of personal text messages with secure multi-party computation. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Edward A. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 3752–3764, 2019.

[50] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721. ACM, 2018.

[51] Ronald L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Preprint available at http://people.csail.mit.edu/rivest/Rivest- commitment.pdf, 1999.

[52] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, volume 62, pages 98–105, 1998.

[53] Rafael Tonicelli, Anderson C. A. Nascimento, Rafael Dowsley, Jörn Müller-Quade, Hideki Imai, Goichiro Hanaoka, and Akira Otsuka. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *International Journal of Information Security*, 14(1):73–84, 2015.

[54] Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. Automatic detection of cyberbullying in social media text. *PloS one*, 13(10):e0203794, 2018.

[55] Thijs Veugen, Frank Blom, Sebastiaan JA de Hoogh, and Zekeriya Erkin. Secure comparison protocols in the semi-honest model. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1217–1228, 2015.

[56] Benjamin Weggenmann and Florian Kerschbaum. SynTF: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 305–314, 2018.