arXiv:2508.09665v1 [cs.CR] 13 Aug 2025

SOCIAL-SENSOR IDENTITY CLONING DETECTION USING WEAKLY SUPERVISED DEEP FOREST AND CRYPTOGRAPHIC AUTHENTICATION

Ahmed Alharbi
College of Computer Science and Engineering,
Taibah University, Medina, Saudi Arabia
email:atharbi@taibahu.edu.sa

Hai Dong, Xun Yi
School of Computing Technologies
RMIT University, Melbourne, Australia
email:{hai.dong,xun.yi}@rmit.edu.au

ABSTRACT

Recent years have witnessed a rising trend in social-sensor cloud identity cloning incidents. However, existing approaches suffer from unsatisfactory performance, a lack of solutions for detecting duplicated accounts, and a lack of large-scale evaluations on real-world datasets. We introduce a novel method for detecting identity cloning in social-sensor cloud service providers. Our proposed technique consists of two primary components: 1) a similar identity detection method and 2) a cryptography-based authentication protocol. Initially, we developed a weakly supervised deep forest model to identify similar identities using non-privacy-sensitive user profile features provided by the service. Subsequently, we designed a cryptography-based authentication protocol to verify whether similar identities were generated by the same provider. Our extensive experiments on a large real-world dataset demonstrate the feasibility and superior performance of our technique compared to current state-of-the-art identity clone detection methods.

***K*eywords** Identity cloning detection · Social-sensor cloud services · Non-privacy-sensitive user profile features · Weakly supervised deep forest model · Cryptography-based authentication

1 Introduction

Social sensing is a paradigm that allows multiple *social sensors* (e.g., humans with smart devices) to collect data [39]. This sensed data is called *social-sensor data* that is in a variety of forms (e.g., texts, images, etc.) and stored in *social-sensor clouds* (i.e., social media platforms) [3, 4]. Examples of social-sensor data include Twitter posts and Facebook status. Social-sensor clouds can be viewed as an essential open channel for social-sensor cloud users to share their personal experiences about special events, such as accidents and public events [39], [3]. Social-sensors can publish thousands or even millions of posts across social-sensor clouds. This type of data can be abstracted as *social-sensor cloud services* (SocSen services) with *functional* (e.g., location, time, etc.) and *non-functional* aspects (e.g., trust, resolution, etc.) [3, 4]. It can portray unique events from a variety of perspectives, including where, when, and what [2].

The prosperity of social-sensor clouds has attracted the attention of cybercriminals in recent years. These cybercriminals frequently attempt to exploit the identities of *SocSen service providers* (i.e., users who publish social-sensor data from social media platforms) to deceive consumers in a number of various ways [27, 36]. Identity cloning is a way to attack the identities of SocSen service providers, in which an attacker creates a fake profile on a social-sensor cloud based on the identity information of a provider. Identity cloning is in two major forms: *single-site identity cloning* and *cross-site identity cloning* [11]. The former applies to a scenario in which the identity of a SocSen service provider is cloned and

registered in the same social-sensor cloud by an intruder, while the latter refers to a scenario where the identity of a SocSen service provider is copied from a cloud to another. Our project focuses on the detection of single-site identity cloning. In recent years, we have noticed a number of famous cases related to single-site identity cloning. For example, Facebook CEO Mark Zuckerberg's Facebook account was cloned and utilized in a financial fraud[1]. Additionally, Russian President Vladimir Putin's cloned Twitter account had attracted over 1 million followers[2].

Most social-sensor clouds do not have mechanisms for automated identity cloning detection. For instance, platforms like Instagram and Twitter only conduct identity cloning investigations after receiving complaints from end-users[3][4]. Additionally, many existing identity cloning detection techniques depend on a mix of privacy-sensitive (e.g., full name, date of birth) and non-privacy-sensitive user profile features (e.g., screen name, profile descriptions) [20, 33, 34]. However, these methods are inapplicable to most third-party websites and applications for detecting cloned identities. This is because most third-party platforms that rely on social media APIs for user authentication (i.e., social login) are restricted from accessing privacy-sensitive user data. On the other hand, non-privacy-sensitive user profile information is generally accessible through APIs to third-party platforms. Therefore, there is a strong need to explore new techniques for identity cloning detection that rely exclusively on non-privacy-sensitive user profile features.

Further, most existing identity cloning detection techniques [20, 24, 32] rely on relatively simple metrics, such as Jaro–Winkler distance and Term Frequency — Inverse Document Frequency (TF-IDF) based cosine similarity, for inter-identity feature similarity measure. These metrics rely on word frequency or character distance and *cannot fully capture the semantics of words and strings*. For instance, the (gender-based) similarity of *king* and *man* cannot be measured using the previous metrics. Deep learning (DL) techniques, such as Bidirectional Encoder Representations from Transformers (BERT), provide embeddings that enable us to create more than one representation for a word based on its context of usage [19]. In this regard, DL can capture the semantics of words and strings. In addition, existing identity cloning detection techniques employ conventional machine learning [24]. Conventional machine learning uses shallow architectures and is incapable of handling large-scale data sources. In this context, DL has been acknowledged as a powerful tool for large data processing and analytics in a variety of domains [37, 47, 46, 42, 31, 29, 21]. DL architecture can transform and represent information at multiple stages.

Against the issues above, we previously proposed 1) an unsupervised identity cloning detection approach based on non-privacy-sensitive user profile features [7] and 2) a deep learning model for identity cloning detection [8, 9]. However, our previous works and the existing approaches still suffer from the following major limitations:

- *Unsatisfactory performance.* The performance of the existing identity cloning detection techniques still has sufficient space for improvement.
- *Lack of solutions for duplicated accounts.* Most of the existing works only focus on detecting similar identities and assume that all of them are cloned. However, it is quite common that many providers create more than one similar identity in a social-sensor cloud using different emails. This kind of identities is also termed as **duplicate accounts**. These identities are obviously not cloned/fake ones created by attackers.
- *Lack of large-scale evaluations on real-world datasets*. The existing techniques were only tested in simulated or small-scale datasets.

To address the aforementioned limitations, we propose a novel technique for SocSen service providers' identity cloning detection based on non-privacy-sensitive user features. Our proposed technique comprises two main components, namely, 1) a similar identity detection approach to detect a pair of accounts that might contain a cloned account and its victim and 2) a cryptography-based cloned identity authentication protocol to validate whether a pair of identified accounts were created by an identical provider. Our main contributions are summarised below.

– We designed a weakly supervised deep forest model to further improve the similar identity detection performance. First, we trained a deep forest classifier to predict the weak labels. Then we used the generated weak labels to train the deep forest classifier.

– We designed a cryptography-based authentication protocol to validate whether or not a pair of identified similar accounts contain a cloned account. This protocol requests the newer account from an account pair to decrypt two random messages respectively encrypted by the newer and older accounts' public keys for authentication. This is expected to identify the most common noises during cloned account detection, which is that a SocSen service provider creates duplicate accounts in a social-sensor cloud.

[1]https://www.nytimes.com/2018/04/25/technology/fake-mark-zuckerberg-facebook.html

[2]https://www.abc.net.au/news/2018-11-29/twitter-suspends-account-impersonating-vladimir-putin/10569064

[3]https://help.twitter.com/en/rules-and-policies/twitter-impersonation-policy

[4]https://help.instagram.com/446663175382270

– We conduct a more comprehensive evaluation on a large real-world dataset. The dataset contains more than 500K accounts collected from Twitter. The experimental results show that our proposed technique outperforms the state-of-the-art identity cloning detection techniques on precision and F1-score.

The remainder of the paper is structured as follows. Section 2 reviews the current literature in SocSen services, identity cloning detection and identity-based encryption. Section 3 presents the details of the proposed framework. Section 4 explains the experimental design and result analysis of our proposed framework. Section 5 concludes the paper.

2 Related Work

This section analyzes and reviews the current literature in SocSen services, identity cloning detection and identity-based encryption.

2.1 Social-Sensor Cloud Services

SocSen services provide simple access to the management of social-sensor cloud data for the development of scene analysis applications. Aamir et al. [3, 4] introduced a collection of SocSen service selection frameworks for user to query scene-related social media images, where SocSen services abstract the functional and non-functional properties of social media images. These frameworks aim to organize social media images spatially, temporally, and contextually. Aamir et al. [2, 5] further proposed a set of SocSen scene analysis service composition models. These models reconsruct tapestry scenes by using the spatio-temporal, textual and image features of the SocSen services.

2.2 Identity Cloning Detection Techniques

A variety of approaches have been proposed for fake user accounts and spammers detection on social media platforms according to a survey [10]. These approaches mostly utilise behavioural features of users (e.g. writing styles, friendship lists, etc) to identify whether or not a user account is trustworthy [35, 48]. However, the behavioural profile features cannot be effectively employed for identity cloning detection, since an adversary can easily mimic the behavioural profile features. This requires us to explore new features that can adequately describe an account pair. Other researchers, on the other hand, employ social network trust links between users. These approaches assume that a spammer/fake user cannot develop an arbitrary number of trusted links with legitimate users [6, 35]. However, in the context of identity cloning, this assumption may not hold true, as adversaries can attempt to clone legitimate users profiles. Thus, cloned accounts can more easily establish trust with legitimate users.

Several approaches have been proposed for detecting social media identity cloning. Vyawahare and Govilkar [43] developed a method to detect fake and cloned profiles by extracting key attributes (e.g., username, friend count, gender) and calculating a similarity index. Profiles with high similarity scores above a threshold are flagged as potential clones. Jethava and Rao [30] introduced a defensive approach to protect against identity cloning. Their method uses similarity measures (e.g., attribute and friend list) to differentiate between cloned and legitimate users. The approach is implemented on the social app server, where friendship requests are checked for authenticity before being approved. Kontaxis et al. [34] introduced a mechanism by which users can ascertain if they have fallen in a cloned identity attack. Jin et al. [32] studied the behaviour of attackers for identity cloning. They presented two approaches to identify suspicious profiles based on profile similarity. Goga et al. [24] proposed an impersonation detection technique. The proposed technique determines whether a pair of accounts are duplicate accounts. It uses a Support Vector Machine classifier to detect impersonation attacks. Kamhoua et al. [33] compared user profiles across social media platforms to prevent cloned identities. A hybrid string-matching similarity technique is employed to determine the extent of profile similarity. Devmane and Rana [20] developed a technique to detect cloned identities in both single- and cross-platforms. They developed a technique to search for similar user accounts.

Most of the current works [20, 33, 34] detect cloned accounts by analysing both privacy-sensitive and non-privacy-sensitive user profile features. Many third-party websites use social media profiles to verify users. These third-party websites are unable to gain access to privacy-sensitive user profile features through social media APIs. Therefore, those techniques cannot be applied by those third-party websites. In addition, all the aforementioned recent works have not considered solutions to distinguish between the cloned and self-duplicated accounts. This inspires us to investigate identity-based encryption for self-duplicated account authentication.

2.3 Identity-Based Cryptography

Identity-based cryptography is a type of asymmetric cryptography. It permits the generation of a public key from an arbitrary string (e.g. user's identity). Unlike asymmetric cryptography, it requires the generation of both public and

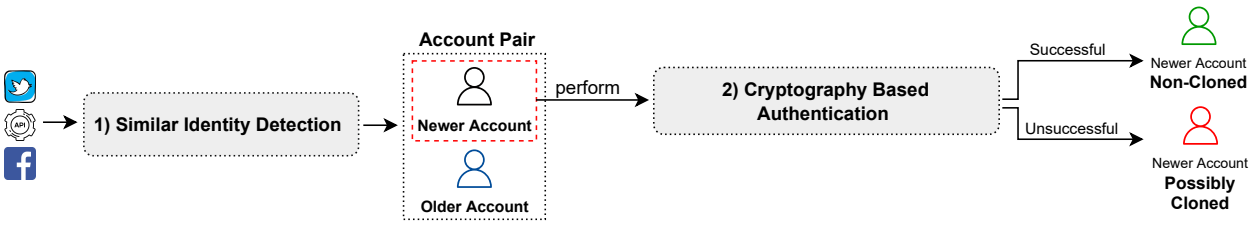


Figure 1: The overview of the proposed solution.

private keys concurrently [41]. Identity-based cryptography simplifies the process of key management. It requires a directory for storing authenticated public system parameters of a Central Authority (CA), which is significantly less cumbersome than keeping a directory for storing all users' public keys [25]. However, identity-based cryptography requires a secure channel for key distribution between the CA and users. Therefore, in our proposed cryptography-based authentication approach, we encrypt the generated private key for a user using a shared session key before the CA distribute the private key.

A variety of identity-based encryption (IBE) schemes have been proposed. Boneh and Franklin [12] proposed an IBE scheme that is both practical and secure. The proposed scheme is indistinguishably safe against attacks that use adaptively selected ciphertexts (i.e. IND-ID-CCA secure). Cocks [16] proposed an IBE scheme using quadratic residues. However, the proposed solution lacks formal security proofs and is very inefficient in terms of bandwidth needs. Sahai and Waters [40] proposed a fuzzy IBE scheme. The proposed scheme permits the encryption of data utilising biometric input as a public key. We utilise the Boneh-Franklin IBE scheme in our proposed cryptography-based authentication approach. The Boneh-Franklin IBE scheme is provably secure and efficient [25]. The result is based on novel assumptions about the difficulty of issues in specific elliptic curve groups.

3 Solution

This section presents a comprehensive overview of the proposed technique and its key components.

3.1 Solution Overview

We propose a SocSen service provider identity cloning detection technique. The proposed approach comprises two main components, namely, 1) a similar identity detection approach and 2) a cryptography-based authentication protocol. Figure 1 shows the overview of the proposed technique. Given a set of social media (e.g., Twitter, Instagram, etc.) accounts[5], we aim to detect if there are possible identity cloning attacks. Firstly, we employ the similar identity detection approach to detect similar account pairs. The proposed similar identity detection is based on weakly-supervised learning. Once an account pair is detected, we perform a cryptography-based authentication process to the newer account in the account pair. The primary goal of this component is to investigate if the account pair is created by the same user. The cryptography-based authentication generates a binary result. If the authentication is successful, it validates that the newer account is not cloned; otherwise the account pair is determined to be highly likely cloned.

3.2 Similar Identity Detection

We introduce an overview of our proposed similar identity detection approach and its key components.

3.2.1 System Design

The proposed similar identity detection approach is illustrated in Figure 2. It comprises four main components: 1) graph construction (GC); 2) an account pair feature representation; 3) a multi-view account representation and 4) a prediction model. First, the GC seeks to construct an undirected graph from a given set of social media users to locate similar accounts pairs. Second, we extract two categories of non-privacy-sensitive user features between the pair of accounts. Third, we create a multi-view account representation from multiple non-privacy-sensitive views for each account in the account pair. The account pair feature representation and the multi-view account representation are then concatenated. Next, we aim to verdict whether the pair of accounts possibly consist of a cloned account and a victim account based on

[5]We use the terms *SocSen service providers* and *social media users* interchangeably throughout the paper unless specified otherwise.

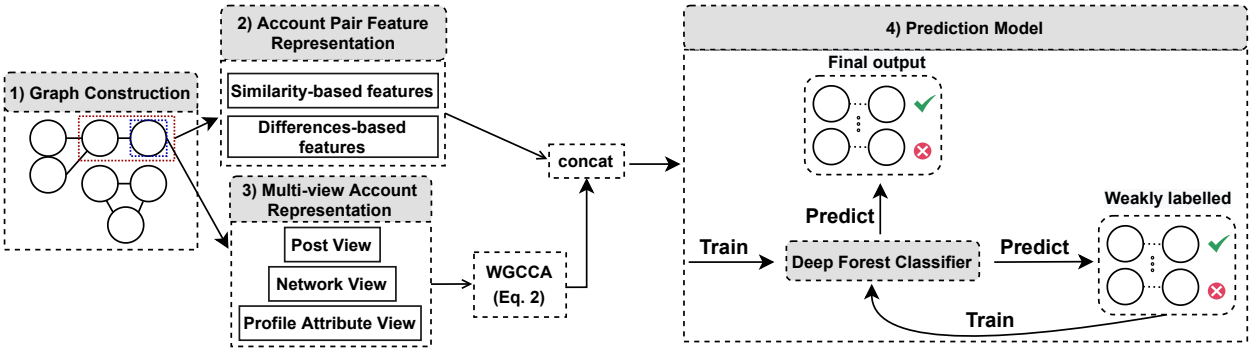


Figure 2: The workflow of the similar identity detection technique

a prediction model. In this regard, we first train a DF model to generate weak labels. Once we generate the weak labels, we train another DF model using the weak labels. The DF model will eventually make the prediction. The following subsections depict these four components in detail.

3.2.2 Graph Construction

Our goal is to build an undirected graph connecting among social media accounts, in which each pair of mutually connected nodes indicates a victim account and a possibility cloned account. The username or screen name of a cloned account is highly likely to be same as or similar to the victim account [24]. As a result, this graph links nodes according to the similarity of the username and screen name. We connect two nodes through an edge only if the similarity score of their username or screen name is higher than a predefined threshold δ. This graph can generate all potential account pairs (a victim account and its clone) in the dataset with a proper threshold (δ) value. We elaborate the procedure of choosing a suitable δ value in the upcoming experiments.

3.2.3 Account Pair Feature Representation

After constructing the undirected graph, we extract a group of non-privacy-sensitive user features for each account pair. All these features can directly be obtained or calculated based on the information extracted from the APIs of mainstream social media platforms, such as Twitter and Facebook. We categorize these features into similarity-based and difference-based features.

(1) **Similarity-based features**

The similarity-based features compare the textual similarity of certain non-privacy-sensitive features (e.g., usernames, screen names, descriptions and locations) between two accounts in each pair. We describe the procedure for computing the similarity-based features as follows.

Screen name, username and location similarity: The Jaro–Winkler string similarity (JS) is one of most effective name similarity metric [15, 17]. Therefore, we use JS to measure the textual similarity between two screen names, usernames or locations.

Description similarity: A user account often contains a brief textual description of the user's affiliations, interests, occupations, etc. We compute the description similarity between a pair of accounts. First, we pre-process the description text by removing stop words and punctuation and converting it to lowercase. We then convert the text into vectors by using the term frequency-inverse document frequency (TF-IDF) metric. Next, we calculate the similarity score between the pair of accounts using cosine similarity.

(2) **Differences-based features**

The differences-based features are utilised to compare the general profile features (e.g., posts counts, friends counts, etc.) of each account pair. We assume that an account pair that possibly contains a cloned account and its victim has a high difference in these general profile features based on the research outcomes of [24]. For instance, a high difference score of the tweet counts between two similar accounts can possibly indicate a victim account and its replica. Altogether, we adopt 10 features across the two aforementioned categories. Table 1 summarizes all the features used for an account pair.

Table 1: Account pair features and descriptions

Feature category	No.	Features	Description
Similarity-based features	1	Screen name similarity	The similarity score of screen name between two accounts.
	2	Location similarity	The similarity score of locations between two accounts.
	3	Username similarity	The similarity score of usernames between two accounts.
	4	Description similarity	The similarity score of descriptions between two accounts.
	5	Followers Ratio	The ratio of the number of followers between two accounts.
Differences-based features	6	Followers differences	The account difference in the number of followers.
	7	Account age differences	The account difference in account age.
	8	Tweets differences	The account difference in the number of tweets.
	9	Favorite differences	The account difference in the number of favorite counts.
	10	Friends differences	The account difference in the number of friends.

3.2.4 Multi-view Account Representation

We aim to create a multi-view account representation for each account in an account pair. The multi-view account representation combines a variety of views that resemble the non-privacy-sensitive profile features of the account. Here we model three views for a user account: 1) post, 2) network and 3) profile attribute views. These views can accurately represent an account, which are likely mimicked by adversaries. Next, we employ weighted generalized canonical correlation analysis (wGCCA) to learn a single embedding from the aforementioned views. This method is effective in capturing the shared structure across multiple views while maximizing the correlation between them [28]. The views are explained below.

(1) **Post View**

For each account in the undirected graph, we extract the pre-trained language representation of the account's posts. We obtain the vector-space representations of user posts using Sentence-BERT (SBERT) [38], which is a variation of the bidirectional encoder representations from transformers (BERT) [19]. We gathered n publicly available posts for each account u, represented as $T = (t_1, ..., t_n)$. Each post $t_i (i \in 1, .., n)$ is represented by the SBERT representation. First, post t_i is tokenized into a list of words w_i. Then special markers [CLS] and [SEP] are added to mark the beginning and ending of a sentence respectively. The BERT layer uses a set of tokenized words to embed fixed-length sentences. The pooling layer then generates t representations using mean aggregation, which outperforms max and CLS aggregation [38]. The output dimension of each post is 385, which is BERT's default output size. Finally, we compute the mean of all posts' representation T for the user account.

(2) **Network View**

A network of user accounts is a cluster of individuals who engage in a social graph. Social media users can engage in various ways, such as befriending, posting, etc. Here we employ two kinds of engaging networks: friend and follower networks. The friend network refers to the accounts (e.g., a superstar or friend) followed by an account and thus these accounts' posts appear on the account's feed. The follower network refers to the accounts that follow another account and those accounts can see this account's posts on their feed. We use Node2Vec to learn the network representation. Node2Vec is well recognized as an efficient technique for network representation [26]. It uses a biased random walk to maximise the log probability of accounts that share an edge.

(3) **Profile Attribute View**

We use 12 public profile attributes to generate the final view of each account. These public profile attributes are all non-privacy-sensitive user features that can portray the behaviours and activities of an account. For instance, the post count may reflect the activity of the account, while the follower count may indicate the reputation of the account. The 12 public profile attributes are presented in Table 2.

Embedding Learning Model

The previously mentioned views may provide comprehensive knowledge that can assist in detecting cloned accounts. Employing each view individually might result in knowledge gaps. One strategy is to concatenate all the previously mentioned views. Nevertheless, the view concatenation may generate a large account representation and cause overfitting

Table 2: The used non-privacy-sensitive user features for the profile attributes view and their descriptions.

No.	Features	Description
1	Follower count	The amount of the user account's followers.
2	Favorite count	The amount of the user account's liked tweets.
3	Tweet count	The amount of user account's posts/tweets.
4	Friend count	The amount of the user account's friends.
5	List count	The amount of public lists of which the account is a member.
6	Account age	The age of user account since registration.
7	Profile background	A binary value indicating if the user account's backdrop or theme has been updated.
8	Screen name length	The user account's screen name length.
9	Profile image	A binary value indicating whether the user account has a uploaded profile image or a default image.
10	Description length	The user account's description length.
11	Has profile description	A binary value indicating whether or not the user account's profile includes a description.
12	Profile URL	A binary value indicating whether or not the user account's profile includes a URL.

with limited training datasets. It may also omit valuable information, as these views might have unique statistical features [45]. For example, each view might have different data patterns (e.g., skewness, uniform, etc.). Accordingly, we employ an alternative approach called generalised canonical correlation analysis (GCCA). GCCA learns a single embedding from a set of views. GCCA has many variants. Here we employ Carroll's GCCA, since it is based on a computationally simple and efficient eigenequation [14]. The objective of GCCA can be formulated as follows.

$$arg \min_{G_i, U_i} \sum_i \parallel G - X_i U_i \parallel_F^2 \qquad s.t. G'G = I \tag{1}$$

where $X_i \in \mathbb{R}^{n \times d_i}$ is the i^{th} view's data matrix, $G \in \mathbb{R}^{n \times k}$ includes all learned account embedding and $U_i \in \mathbb{R}^{d_i \times k}$ maps the latent space with the observed view i. Each view, however, may contain varying degrees of knowledge required to detect cloned accounts. Thus, we use weighted GCCA (wGCCA) that adds weight w_i to each of the views i in Equation 1, as shown in Equation 2:

$$arg \min_{G_i, U_i} w_i \sum_i \parallel G - X_i U_i \parallel_F^2 \;\; s.t. G'G = I, w_i \geq 0 \tag{2}$$

where w_i is the weight of a view, which indicates its importance. The columns of G are the eigenvectors of $\sum_i w_i X_i (X_i{}' X_i)^{-1} X_i{}'$. The solution is $U_i = (X_i{}' X_i)^{-1} X_i{}' G$.

3.2.5 Prediction Model

We aim to predict if a pair of similar accounts possibly comprise a cloned account and its victim. We first train a Deep Forest (DF) model to generate weak labels. Once the weak labels are generated, we employ them to train a new DF model to make the prediction.

(1) **Deep Forest Model**

We first concatenate the account pair feature representation and the multi-view account representation to generate the final accounts pair representation (A_i).

$$y = classifer(concat(F, wgcca) \tag{3}$$

where F is a feature vector F ($\in \mathbb{R}^{10}$), in which each feature F_i in F represents an augmented feature obtained from the account pair feature representation and $wgcca$ is the multi-view account representation embedding.

We utilise the DF model to predict if an account pair possibly includes a cloned account and its target account. The DF model is an ensemble decision tree framework that works effectively with less data and fewer hyperparameters. The structure of the DF model follows a cascaded pattern, in which the input in each layer is a concatenation of the feature vectors produced by the previous layer [51].

Diversity is strongly advised for constructing an ensemble model [49]. Hence, we employ two random forests (RF), extremely randomized trees (ERT) and logistic regression (LR) in the proposed DF model. Figure 3 depicts the proposed DF architecture, which contains multiple self-adaptive layers. We elaborate on the procedure of choosing the proposed DF architecture in Sec. 4.5.2. The RF model is an ensemble classifier that integrates a large number of decision trees and averages their results to enhance the prediction performance [13]. The ERT model is similar to the RF model.

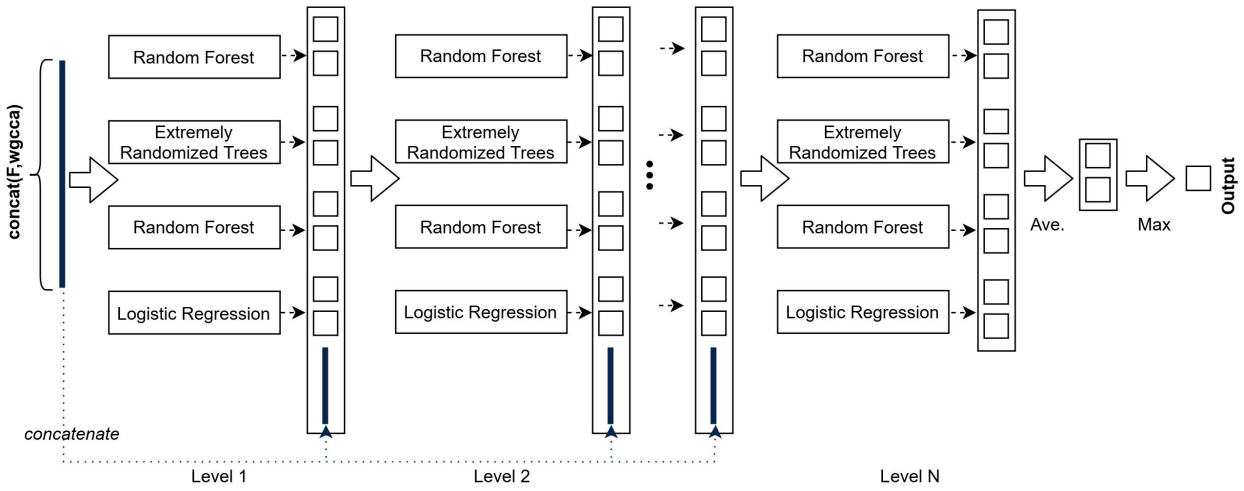


Figure 3: The proposed DF model architecture.

However, they differ in how splits are determined. A RF can be divided in terms of trees, while the ERT splits randomly [23].

The model is fed with the concatenated feature vector $concat(F, wgcca)$. Herein, we aim to obtain a binary value y indicating whether or not an account pair highly likely contains a cloned account and its victim. In this regard, each of the employed models (i.e., 2 RFs, ERT, LR) generates a binary value separately. Thus, the input for the next layer is a set of $8 (= 2 \times 4)$ augmented features. The class vector is generated using k-fold cross-validation $(k = 5)$. This generates $k - 1$ class vectors, which are then averaged to generate the final class vector as the augmented features for the cascade's subsequent layer. After adding one extra layer, the validation set is used to measure the prediction performance of the whole cascade. If there is no substantial improvement in class prediction performance, the cascade growth is immediately terminated. As a result, it can be claimed that the number of cascaded layers is self-determined [51].

(2) **Weak Label Generation**

Weakly supervised learning is an emerging research area. It aims to leverage limited, noisy, or imprecise inputs to produce labels for large-scale training data in a supervised learning setting. Weakly supervised learning can take three different shapes: incomplete, inexact and inaccurate supervision [50]. Incomplete supervision refers to the situation in which only a fraction of training data is labelled, such as active learning and semi-supervised learning. Inexact supervision arises from the fact that the training data is provided with only coarse-grained labels.

Inaccurate supervision (i.e. weak labels) refers to a scenario in which the training data is not always accurate. In other words, some labelled data may contain errors. Weak labels can be utilised with the notion that they are imprecise but may be used to build a robust prediction model [50]. Weak labels may reduce the cost of building ground truth datasets that are prohibitively expensive or impractical. It is common that large-scale identity cloning datasets often contain noisy samples (e.g., duplicated accounts). In addition, the ground truth is usually difficult to be obtained for such large-scale datasets since social media providers do not usually disclose it. Consequently, creating datasets of weak labels may benefit the training of DL models that usually rely on large datasets. For social media identity cloning, such weak labels can be obtained from training a DL classifier.

Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$ denote a set of n account pairs, with $\mathcal{X} = \{x_i\}_{i=1}^{n}$ denoting the accounts pair representation and $\mathcal{Y} = \{y_i\}_{i=1}^{n} \subset \{0, 1\}^n$ denoting the corresponding clean labels indicating whether or not the account pair contains a cloned account and its victim. For the weak labels, we predict the weak labels using the DF model described in the previous section (Sec 3.2.5). The weak labels are denoted as $\hat{\mathcal{D}} = \{\hat{x_j}, \hat{y_j}\}_{j=1}^{N}$ where $\hat{\mathcal{X}} = \{\hat{x_j}\}_{j=1}^{N}$ denotes the N unlabeled accounts pair representation and $\hat{\mathcal{Y}} = \{\hat{y_j}\}_{j=1}^{N}$ is the set of weak labels predicted from the DF model.

(3) **Model Training**

Given a set of samples $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$ with ground truth (clean) labels and a set of weak labels $\hat{\mathcal{D}} = \{\hat{x_j}, \hat{y_j}\}_{j=1}^{N}$, the DF model described in the previous section aims to predict a sample x. The label of a sample is predicted in Equation 3.

3.3 Cryptography Based Authentication

We present a cryptography-based authentication protocol to verify if an account pair detected by the similar identify detection approach is created by the same user. Our protocol is built on the assumption that a newer account in a pair of similar accounts is more likely a cloned account according to the existing study. This assumption is supported by research showing that attackers usually choose well-established accounts as those accounts have more credibility and trust within the social network[22]. Within this protocol, an authentication agent (AA) encrypts two random messages respectively using the older and newer accounts' identities and requests the newer account to successfully decrypt these encrypted messages for authentication. In the remaining of the subsection, we first introduce the preliminaries and describe the system model; next, we explain the proposed framework architecture and the adopted algorithms; finally, we analyze the convenient attacks during the authentication process and analyze how these attacks can be resolved with our solution.

3.3.1 Preliminaries

This section provides a brief background introduction about Bilinear Maps, Identity-Based Encryption (IBE) and Advanced Encryption Standard (AES), which are the primary techniques from which the proposed solution is derived.
(1) **Bilinear Maps**
Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups with a prime order p. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a map with the following properties:

Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$.

Non-degeneracy: there exist $g_1, g_2 \in \mathbb{G}_1$ such that $e(g_1, g_2) \neq 1$

Computability: for all $a, b \in \mathbb{Z}_p$, an efficient algorithm exists to compute $e(g_1, g_2)$
(2) **Identity-Based Encryption (IBE)**
We adopt the Boneh-Franklin IBE scheme [12] in our proposed solution. The Boneh-Franklin IBE scheme is provably secure and efficient [25]. The scheme consists of the following algorithms:

Setup: The input of the setup algorithm is a security parameter t and it outputs a public parameter (PK) and a master key (MK).

Extract: The inputs of the extract algorithm include the public parameter (PK), the master key (MK) and an identity (ID). It outputs a private key d_{ID} for the identity.

Encrypt: The encryption algorithm takes a randomly created message (M), the public parameter (PK) and the identity (ID) as inputs. It outputs the encrypted message (C).

Decrypt: The decryption algorithm takes the ciphertext (C), the public parameter (PK) and an identity (ID) as inputs. It outputs the message (M).
(3) **Advanced Encryption Standard (AES)**
The AES algorithm is a well-known approach for symmetric key encryption. It performs encryption and decryption using a single key. AES uses a 128-bit block size and a 128-bit, 192-bit, or 256-bit key size. The key size for an AES cipher is determined by the number of AES rounds, which is 10, 12, or 14 for 128, 192, or 256-bit keys, respectively. The AES encryption process is operated on a 4×4 byte matrix that is called a state array. Each AES round contains several encryption processing steps [1] which are described below:

SubBytes: A non-linear substitution phase in which each byte is substituted by another using a lookup table.

ShiftRows: Each row of the 4x4 matrix is shifted cyclically by a predefined offset.

MixColumns: A linear mixing operation that mixes the columns.

AddRoundKey: XOR operation is performed between each byte of the state and the subkey, which is obtained from the main key using Rijndael's key schedule [18] that converts a short key into a series of independent round keys.

3.3.2 System Design

Our proposed framework is an extension of the Boneh-Franklin IBE scheme [12] that assumes that the private key generator (PKG) (i.e. AA) is a fully trusted entity. The major difference between the Boneh-Franklin IBE scheme and our proposed framework is that the former has only one AA and the latter has multiple AAs. Besides, both the AA and CS reside in the cloud side and are fully managed by the social-sensor cloud provider, which can be assumed as fully trusted [44]. Our proposed framework for cryptography-based authentication is shown in Figure 4, including:

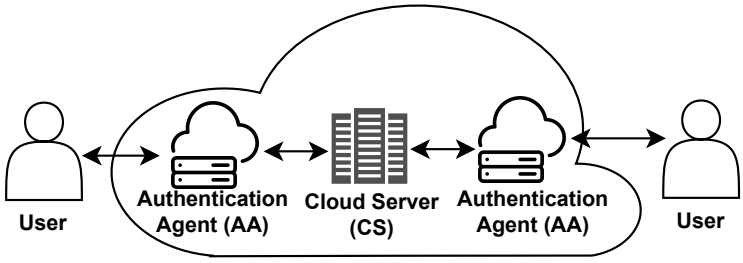


Figure 4: System model

Authentication Agent (AA): Each AA is a fully trusted entity that is responsible for generating a private key for a new account. It is also responsible for delivering the private key to the new account through secure channels and encrypting a random message and decrypting an account's response for account authentication.

Cloud Server (CS): The CS is also a fully trusted entity. It offers a cloud-based data storage solution. Social-sensor cloud providers deploy CS to facilitate the storage of detected similar account pairs and their authentication status.

Users: The users obtain their private keys via registration. They are also required to authenticate themselves when there is are existing similar accounts.

3.3.3 The Authentication Framework Architecture

The proposed authentication protocol consists of two main parts: 1) key generation process; 2) authentication process.

Key Generation Process Figure 5 shows the sequence diagram of the key generation process.

Register a new user account: Every time when a new account is registered, the system will send the account information (i.e. the username) to an AA to receive a private key. During this process, this new account generates a session key. The session key is encrypted using the AA's public key (i.e. the AA's ID) with the Encrypt algorithm defined in Section 3.3.4. Finally, the encrypted session key and the user's identity are sent to the AA to request a private key.

Send encrypted private key: Once the AA receives a private key request containing an encrypted session key and a username, it decrypts the session key by running the Decrypt algorithm depicted in Section 3.3.4. The AA then generates a private key for this registered account by running the Extract algorithm in Section 3.3.4. Next, the AA encrypts the private key using the session key by applying the AES-Enc$_{PK}$ algorithm in Section 3.3.4. Finally, the AA sends the encrypted private key to the registered account.

Decrypt private key: Once the registered account receives the encrypted private key, the registered account decrypts the private key by running the AES-Dec$_{PK}$ algorithm described in Section 3.3.4.

Authentication Process Figure 6 shows the sequence diagram of the authentication process.

Send a similar account pair: Each time the CS detects a similar account pair whose authentication status is null, it will send the account pair information to an AA. The information includes the username and registration date of each account in the account pair.

Send encrypted message: Once the AA receives the information of a similar account pair, it checks the registration date of the account pair. The AA then sends two encrypted messages to the newer account to request it to decrypt the messages for the authentication purpose. The encrypted messages are two random messages respectively encrypted using the older and newer accounts' public keys (i.e. their usernames). The AA will run the Encrypt algorithm to encrypt the random messages. The encrypted messages can only be decrypted using their corresponding private keys.

Decrypt the encrypted message: Once the newer account receives the encrypted messages, the account user is required to decrypt the encrypted messages respectively with the older and newer accounts' private keys using the Decrypt algorithm. Here the responses can be categorized into two basic types: 1) The decryption is successful (i.e. both the messages are successfully decrypted), which indicates that the owner of the newer account owns the older account's private key and the account pair is created by the same user. 2) The decryption is unsuccessful (i.e. none or either of the messages are successfully decrpyted), which indicates that the newer account is possibly the clone of the older account or an attacker who steals the private key of the newer or older account.

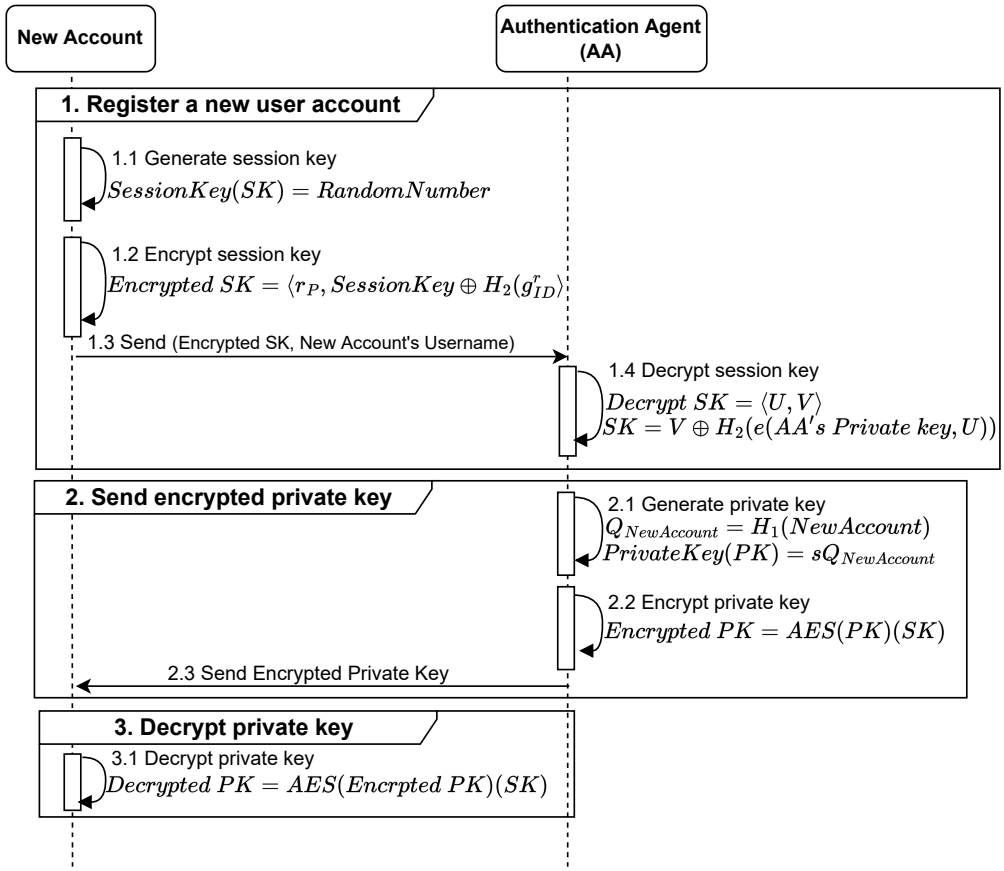


Figure 5: A sequence diagram of the key generation process.

Send encrypted response: The response of the newer account is encrypted with the AA's ID using the Encrypt algorithm and sent to the AA.

Decrypt the encrypted response: Once the AA receives the encrypted response from the newer account, the AA decrypts this response with its private key by running the Decrypt algorithm.

Send authentication response: The AA will send the decrypted response to the CS to update their authentication status (i.e. successful or unsuccessful).

Eventually, all the unsuccessfully authenticated account pairs are suspected to highly likely contain a cloned account and need further investigation.

3.3.4 System Definition

We explain the algorithms adopted by the framework.

Setup: This algorithm only runs on the side of the AA. The AA selects a bilinear pairing $e : G \times G \rightarrow H$, where G and H are two cyclic groups of prime order p, with the size of q, where q is prime power. The AA then selects a random value s, such that $s \in Z_q^*$ and calculates $P_{pub} = sP$, where P and sP are point on elliptic curve. Next, the AA employs the cryptography hash function $H_1 : \{0,1\}^* \rightarrow G$, which allows the AA to map an identity ID to an elliptic curve point. The AA also needs a hash function H_2 such that $H_2 : H \rightarrow \{0,1\}^n$ to encrypt a message with size n. Finally, the system master public key (MPK) is $MPK = \langle e, n, G, H, P, P_{pub}, H_1, H_2 \rangle$ and the system master private key (MSK) is $MSK = s \in Z_q^*$ for the AA. The AA uses the MSK to generate a private key based on a user's $username$.

Extract: This algorithm also runs at the side of the AA only. A private key request (Req_{PK}) containing a user's $(username)$ and an encrypted session key using the AA's ID is sent to the AA. Once receiving the Req_{PK}, the AA maps the user's $username$ to a point P on the elliptic curve E as follows: $Q_{ID} = H_1(ID) \in G$, where Q_{ID} is the

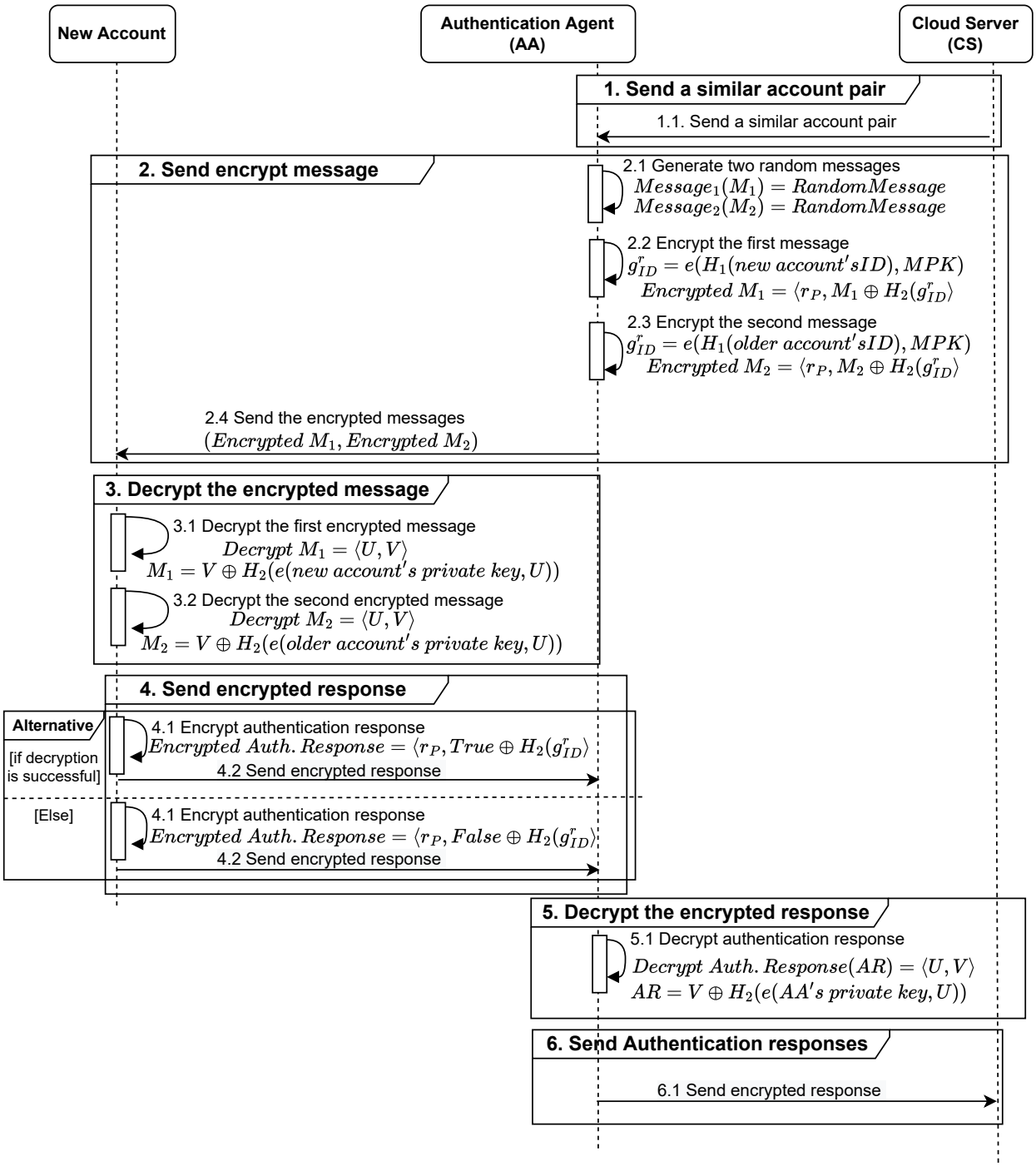


Figure 6: A sequence diagram of the authentication process.

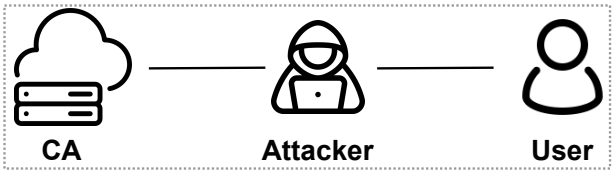


Figure 7: Secure channel attack model

result of mapping the user's $username$ to a point P. The user's private key d_{ID} is set to $d_{ID} = sQ_{ID}$, where s is the master private key of the AA.

AES-Enc$_{PK}$: After the AA generates a private key for a given user, the AA uses Advanced Encryption Standard (AES) [1] to encrypt the user's private key using the shared session key. The AA then sends the encrypted private key to this user.

AES-Dec$_{PK}$: After the user receives his/her private key from the AA, the user uses AES to decrypt his/her private key using the shared session key.

Encrypt: The encryption algorithm selects a random integer $r \in Z_q^*$. It then uses the recipient's $username$ to calculate $Q_{ID} = H_1(ID) \in G$. Finally, it computes a ciphertext $C = \langle r_P, M \oplus H_2(g_{ID}^r \rangle$, where $g_{ID} = e(Q_{ID}, MPK) \in H$.

Decrypt: On receiving the ciphertext C ($C = \langle U, V \rangle$), where U and V are the components of ciphertext C, the recipient uses his/her private key d_{ID} to decrypt C. The decrypted message M is calculated by $M = V \oplus H_2(e(d_{ID}, U))$.

3.3.5 Attack Analysis

We demonstrate how the proposed framework is proven to be secure and robust against potential attacks.

Channel Attacks The proposed protocol requires the AA to interact with a user account during the key generation and authentication process. In such cases, an attacker can steal the information transmitted between the AA and the user account shown in Figure 7.

Prevention of Channel Attacks We prevent such an attack by encrypting the data transmitted between the user account and the AA in the following interactions:

Private key generation: With the proposed protocol, each account first generates a session key that is only shared between the account and the AA. The session key is encrypted using the AA's ID (Step 1 in Section 3.3.3). The AA then generates a private key for the account. Next, the AA encrypts the generated private key for the account using the shared session key (Step 2 in Section 3.3.3). By encrypting the private key, this protocol provides a secure channel between the AA and the account for private key transmission.

Authentication: The random messages generated by the AA for authentication are respectively encrypted using the newer and older accounts' usernames (Step 2 in Section 3.3.3). The newer account is required to perform more secure double-key authentication (i.e. decrypting both the messages respectively using the newer and older accounts' private keys) to prove its identity. In addition, the authentication response of each account is encrypted using the AA's ID (Step 4 in Section 3.3.3). By encrypting the authentication messages and response, this protocol provides a secure channel for transmitting the authentication data between the AA and the account. The formal proof of the encryption and decryption was provided in [12].

4 Evaluation

We conducted a set of experiments to verify the effectiveness of the proposed solution. The experiments will answer the following four key questions:

RQ1: What are the impact of the components and the optimal hyperparameter values of the similar identity detection approach?

RQ2: Does our solution outperform the state-of-the-art cloned identity detection approaches and other machine learning and deep learning-based models?

RQ3: What is the required time efficiency of the cryptography-based authentication protocol?

RQ4: What is the required key storage capacity of the cryptography-based authentication protocol?

4.1 Experimental Environment

All the experiments were conducted on a Ubuntu Server 20.04 LTS with Intel Core i5 1.80 GHz CPU and 16 GB RAM. Furthermore, all the candidate models were implemented in Python. We extracted the pre-trained language representations utilised in similar identity detection using the SBERT package[6]. Additionally, we extracted the Node2Vec representations employed by similar identity detection using the StellarGraph package[7]. We implemented the DL models evaluated using the Python-based Tensorflow[8] library and the other machine learning models evaluated using scikit-learn[9]. For the cryptography-based authentication, we used the Charm[10] framework to implement Boneh-Franklin Identity based encryption. We also used PyCryptodome[11] to implement AES based encryption. Finally, we implemented a prototype of the cryptography-based authentication protocol, which can be downloaded from[12]. In addition, we executed all the machine and DL models 10 times with different random data combinations. The results are presented as a mean of all experiments conducted during the tests.

4.2 Dataset

To the best of our knowledge, Twitter is the only mainstream social media platform that has disclosed a set of cloned accounts[13]. It contains 7,015 possibly cloned accounts and their targets (i.e. another set of 7,015 accounts). The majority of current research, albeit restricted in scope, evaluated proposed methodologies using simulated data. As a result, based on those Twitter accounts, we created a dataset to assess our proposed solution. We retrieved the non-privacy-sensitive user profile features of those accounts through Twitter APIs[14]. Further, most social media platforms claim that fake accounts (including cloned accounts) are a minority. For example, Twitter claims that fake accounts only account for about 5% of the total accounts. Therefore, to mimic a real-world social media environment where cloned profiles are a minority, we randomly collected 500,000 public Twitter user accounts. Eventually, we developed a dataset that comprises a total of 514,030 public Twitter profiles[15].

4.3 Other Methods Evaluated

We evaluated and compared our proposed similar identity detection approach against a comprehensive set of existing identity cloning detection approaches listed below

Basic Profile Similarity (BPS) [32]: This approach examines how much a specific user account and its presumed cloned account overlap using public features and similar friends.

Devmane and Rana [20]: This approach extracts user accounts' names, workplace, images, location, birthday, education, gender, and friends counts. It then compares these extracted features against a set of user accounts.

Goga et al. [24]: This technique extracts different user account features. It extracts user account public features, overlap account friends, overlap of account tweeting and differences between accounts. A linear kernel is then used to train an SVM classifier, which is subsequently used to identify whether a given account has been impersonated.

Kamhoua et al. [33]: This technique evaluates the similarity of friend lists and calculates the similarity of features operating an adjusted similarity measure called Fuzzy-Sim. It examines the following features: name, city, friend list, place of employment age, gender and education. We utilised the same Fuzzy-Sim threshold values (i.e. 0.565 and 0.575) as indicated in the original paper.

NPS-AntiClone [7]: This approach uses only the multi-view account representation of each account in the GC. It then determines the cosine similarity of the two accounts. Then, if the resemblance is more than 0.1, the account pair comprises the cloned account and its related target account.

Zheng et al. [48]: This model is generally used to detect spammers. It utilises 18 different types of features such as profile-related (e.g. follower count) and content-related (e.g. the average count of hashtags). Next, an SVM classifier based on a Radial Basis Function (RBF) kernel is trained to identify if a user account is a spammer user account.

[6]https://github.com/UKPLab/sentence-transformers

[7]https://github.com/stellargraph/stellargraph

[8]https://www.tensorflow.org/

[9]https://scikit-learn.org/stable/

[10]https://github.com/JHUISI/charm

[11]https://github.com/Legrandin/pycryptodome

[12]https://github.com/a7madtalal123/Social-Sensor-Cloud-Service-Provider-Identity-Cloning-Detection

[13]https://impersonation.mpi-sws.org/

[14]https://developer.twitter.com/en/docs

[15]https://www.reuters.com/technology/twitter-estimates-spam-fake-accounts-represent-less-than-5-users-filing-2022-05-02/

Table 3: Values of hyperparameter for ML/DL models

Model	Parameter
ADA	estimators = 100
RF	estimators = 50
MLP	activation = relu, solver = adam
CNN	8 layers, filters = 64 and 8, kernel size = 2 and 1, pool size = 2
DNN	6 layers (300, 250,150,100, 50, 1)
SVM	kernel = linear
KNN	neighbors = 15

DF$_{Clean}$ [8]: This approach is similar to our proposed similar identity detection approach. However, it only uses clean labels to train the DF model.

Furthermore, we evaluated the proposed DF model against the listed below machine learning and DL models to validate the usage of the DF model as the classifier of the cloned identity. The current literature extensively employed the following models in the area of cloned identity detection in social media [10]. These models include LR, RF, Adaboost (ADA), Deep Neural Network (DNN), K nearest neighbours (KNN), Support Vector Machine (SVM), Convolutional Neural Network (CNN) and Multi-layer Perceptron (MLP). Additionally, we evaluated the proposed DF model to variants of its base learners (ERT-based DF (DF$_{ERT}$), LR-based DF (DF$_{LR}$) and DF$_{2RF,2ERT}$) to demonstrate the model structure tuning process.

4.4 Hyperparameter Tuning

We fine-tuned all the hyperparameters of the supervised machine learning and DL models. Table 3 details the values of the hyperparameter utilised to configure the machine learning and DL models. Additionally, all the models are based on the non-privacy-sensitive user features obtained through Twitter APIs.

We configured the hyperparameters of the similar identity detection solution. The GC's optimal δ value is 0.8 according to the experimental results (see Section 4.5.1). We utilised 'all-MiniLM-L6-v2'[16] as the pre-trained model for SBERT. This pre-trained model was trained on millions of paraphrased sentences. SBERT's default dimension for its post representation is 385. For the Node2Vec, we used its default dimension size (128 dimensions) for the network view (i.e. follower and friend network). Additionally, we utilised $q = 2$ as the likelihood of moving away from the source node, $p = 0.5$ as the likelihood of returning to the source node and 15 as the maximum length of a random walk. We normalised all profile attribute views to $[0, 1]$. We set the wGCCA's weights w to $[0.25, 0.5, 0.5, 0.25]$ which were determined in the experiment (see Section 4.5.1).

4.5 Results and Discussion

We used Precision, Recall and F1-Score in our evaluation. Precision is the ratio of accurately predicted account pairs (i.e. a cloned account and its related target), while Recall is the ratio of true account pairs that are accurately detected.

4.5.1 Impact of the components and hyperparameter (RQ1)

Impact of the major components We evaluated the contribution of the key components of our proposed approach by comparing its performance with variants that excluded either the multi-view account representation (wGCCA), account pair feature representation, or weakly supervised labelling. The results demonstrate that integrating all three components significantly enhances detection accuracy, as they complement each other by capturing different aspects of the data. Table 4 presents the detailed comparison results.

Impact of the views To assess the contribution of the individual views in the multi-view account representation, we compared the performance of wGCCA-based detection (without account pair feature representation or weakly supervised labelling) using each view in isolation, as well as their combination. The analysis shows that combining all views—post view, network view, and profile attribute view—substantially improves the model's performance, highlighting the importance of a comprehensive multi-view approach. Table 5 summarizes the comparison results.

[16]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Table 4: Impact of the major components

Model	F1-Score(%)
Our solution (w/o wGCCA)	80.13
Our solution (w/o Account pair feature representation)	78.32
Our solution (w/o Weakly supervised labelling)	79.77
Our solution (Combined)	**82.07**

Table 5: Impact of the views on wGCCA based detection

Model	F1-Score(%)
wGCCA (Profile attributes)	53.17
wGCCA (Posts)	55.17
wGCCA (Network)	61.81
Our wGCCA (All views)	**71.89**

Impact of the threshold of GC We analyzed the impact of the GC threshold (δ), as shown in Figure 8. It can be seen that Recall increases when the δ score was gradually declined from 0.8 to 0.1. This means that, when the similarity score is lowered, the GC located more cloned account and victim account pairs. However, Precision is very low. We attribute that the GC located more irrelevant account pairs. This would remarkably impact the efficiency of the proposed solution. Therefore, we choose 0.8 since it locates most possibly cloned account pairs with a relatively lower number of similar account pairs.

Impact of the wGCCA's weight w We conducted a set of experiments to determine the effect of the wGCCA's weight w on each view (i.e. post, network and profile views). To perform this, we examined several weight combinations (i.e. 0.25, 0.5, and 1) for each view. Each view was given a weight [*post, friend, follower, attribute*]. The top 10 outcomes obtained against various weight combinations are shown in Figure 9. We make the following observations. The F1-Score was increased when the profile attribute view was assigned with lower weights. Next, the post view did not impose much impact on the wGCCA embedding. In contrast, the network view (i.e. friends and followers) had a high impact. The F1-Score was risen when the weight of network view was increased. We finally chose $[0.25, 0.5, 0.5, 0.25]$ as the optimal values of w for each view.

Impact of the DF structure We performed a series of experiments to analyze the impact of the DF structure. To this end, we tested various structures of the DF. Table 6 shows the performance results. Our proposed $\text{DF}_{2RF,ERT,LR}$ (DF_{Clean} [8]) structure achieved 80.82%, 78.76% and 79.77% on Precision, Recall and F1-Score, respectively. Employing a single type of classifier did not produce satisfactory performance results. Our proposed DF structure generates a cascading ensemble of these basic learners (i.e. ERT, LR and RF), in which each cascade consists of the aforesaid basic learners. In this regard, the cascading ensemble models the diversity of learning functions, which improves its generalisation performance and thus ultimately leads to a significant performance improvement.

Impact of the weak label percentage In real scenarios, the amount of ground truth data is usually limited. On the other hand, a large amount of unlabeled data is available. In this experiment, we analyzed the impact of the weak labels percentage on our similar identity detection. To this end, we varied the weak label proportions between 60% and 100%. Figure 10 shows the performance results on those weak label percentages. It can be seen that the 100% weak labels generate the highest F1-Score. This again validates the performance of the weak labels on improving the robustness of the trained model. Accordingly, we trained the DF model purely on all the weakly labelled data regenerated by the same DF model.

4.5.2 Performance of the proposed similar identity detection approach (RQ2)

Table 7 shows the result of the performance comparison. Our proposed approach outperformed all the existing state-of-the-art identity detection approaches in terms of Precision and F1-Score. In addition, our proposed DF model trained on weak labels achieves better performance than the DF model only trained on clean labels on the same metrics. This shows the robustness of the weak label based training. However, the recall rate of the DF drops when being trained on weak labels. In this regard, the weakly labelled training data contain noises that affect the performance of the model on retrieving the number of true cloned account pairs. The techniques proposed by Kamhoua et al. [33] and Devmane and

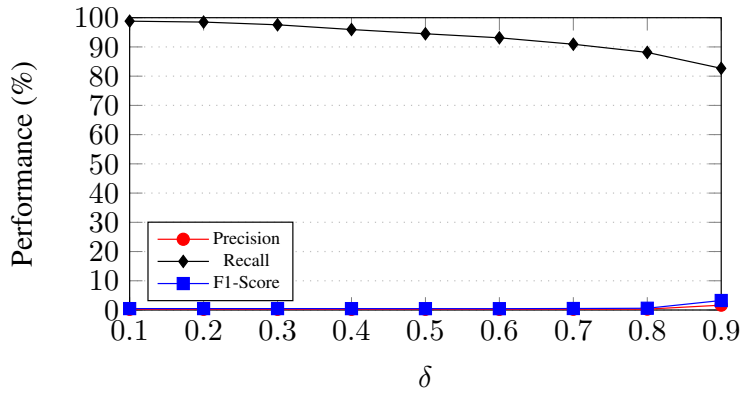


Figure 8: Impact of the threshold of GC

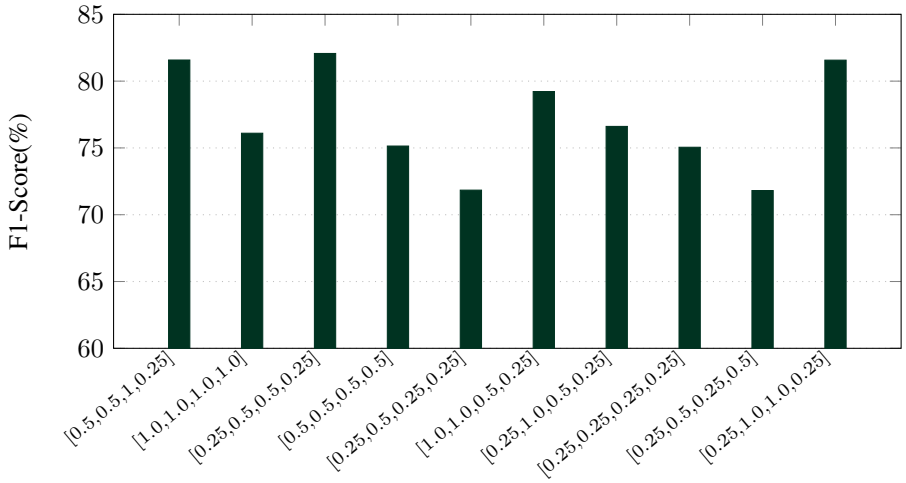


Figure 9: Impact of the wGCCA's weight w

Rana [20] performed poorly since they use simple similarity techniques to calculate the similarity of profile features. They also did not take into account the post and network views of users. BSP [32] is also based on simple profile attribute similarity and shared friends between account pairs. Similarly, Goga et al. [24] employed a conventional similarity method that is based on human-defined measures. Zheng et al. [48]'s method entirely focuses on spammer detection. It detects whether or not an account is a spammer by analysing its spammer behaviour patterns. However, cloned profiles' behaviour is different from spammers. Cloned profiles attempt to replicate the behaviour of genuine profiles and thus are difficult to be detected by spammer detection techniques. The results show that our proposed similar identity detection approach better fits the scenario in which only the non-privacy-sensitive user profile features are employed to detect identity cloning.

Table 8 shows the comparison results between the proposed DF model and the other machine learning and DL models. It can be seen that our DF model notably surpassed all of the other candidate models on all the three metrics. The DF can automatically adjust the model complexity to adapt to the actual problem context. With this feature, the generation of DF models tends to be considerably simpler to handle and the quality of the generated models appear to be more adaptable to the given problem compared with the other models (e.g. DNNs) [51].

4.5.3 Time efficiency (RQ3)

In this experiment, we analyzed the time consumption required by the cryptography-based authentication protocol. Our proposed authentication protocol contains a number of algorithms (see Section 3.3.4). Thus, we evaluated the time cost of each algorithm when being applying to the protocol (see Section 3.3.3). The efficiency of our proposed authentication protocol is shown in Figure 11. Figure 11a shows the time consumption for extracting a private key versus the number of users. It approximately takes around 8s for 500k accounts. In Figure 11b, we can see the time

Table 6: Impact of the DF structure

Model	Precision(%)	Recall(%)	F1-Score(%)
DF_{RF}	83.13	73.35	77.65
DF_{ERT}	49.73	68.98	56.09
DF_{LR}	83.83	30.32	44.49
$DF_{2RF,2ERT}$	**84.26**	72.49	77.89
$DF_{2RF,ERT,LR}$ (DF_{Clean} [8])	80.82	**78.76**	**79.77**

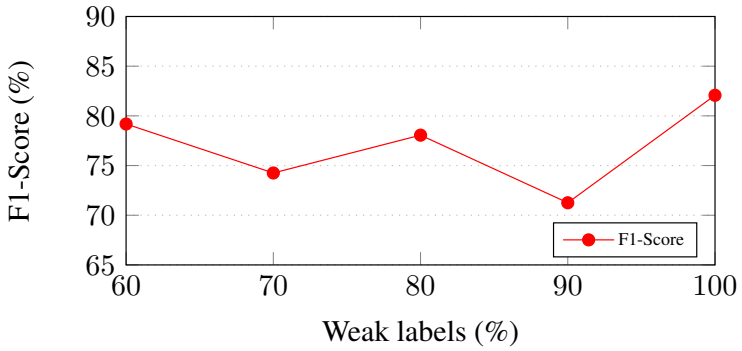


Figure 10: Impact of the weak labels percentage

(approximately 90s for 500k accounts) taken for encrypting the user's private key while in Figure 11c we can see the time (less than 50s for those accounts) consumed for decrypting a user's private key. Figure 11d shows the average encryption time of a session key of a user is around 0.01s. In Figure 11e, the average time taken for decrypting a session key of a user is 0.013s. Figure 11f and g show the encryption and decryption time of random messages. It takes around 0.029s and 0.026s on average to respectively encrypt and decrypt two messages. In Figure 11h and 11i, we can see that the encryption of an account's responses takes 0.015s approximately and its decryption takes 0.012s roughly. Overall, it can be seen that the time costs of all these algorithms are increased linearly along with the rising number of accounts. Thus, it means that the time consumption growth is always kept at a relatively fixed rate. The evaluation results show that our proposed authentication protocol is computationally efficient.

4.5.4 Key storage (RQ4)

We analyzed the storage capacity required by storing the keys for the cryptography-based authentication protocol. The storage of the private and public keys required by authentication in our dataset is shown in Figure 12. It can be seen from Figure 12a that only one private key is required for the newly registered account at registration time. The length of each account's private key is 1024 bits. Similarly, a public key is required for each account and is the account's username in this case, as shown in Figure 12b. Overall, it can be observed that the required key storage capacity is increased linearly. The result of the required key storage capacity shows that our proposed authentication protocol provides very effective key storage solution.

4.6 Summary of Findings

We can make the following conclusions based on the aforementioned experimental analysis: 1) Our proposed similar identity detection approach outperformed the state-of-the-art-identity cloning detection methods. Additionally, it outperformed other machine and DL models (e.g. RF, DNN, etc). 2) Our proposed cryptography-based authentication is proved to be efficient in terms of time efficiency requirement and effective in terms of storage requirement.

Table 7: Comparison with the existing state-of-the-art identity detection techniques.

Method	Precision(%)	Recall(%)	F1-Score(%)
BSP [32]	68.31	75.14	71.56
Devmane and Rana [20]	64.31	77.14	70.15
Goga et al. [24]	65.85	73.74	69.57
Kamhoua et al. [33]	60.17	76.66	67.42
NPS-AntiClone [7]	71.14	67.67	69.36
Zheng et al. [48]	70.75	71.47	71.11
DF_{Clean} [8]	80.82	**78.76**	79.77
Our solution	**91.04**	74.71	**82.07**

Table 8: Comparison with the candidate machine learning and DL models evaluated as the predictor of our proposed similar account detection technique

Model	Precision(%)	Recall(%)	F1-Score(%)
ADA	83.17	41.42	52.47
CNN	89.47	58.65	70.48
DNN	90.87	43.94	58.01
KNN	89.40	49.66	63.61
LR	86.74	22.33	34.34
SVM	85.39	28.71	42.93
MLP	81.51	64.25	71.84
RF	78.36	64.51	70.27
Our solution	**91.04**	**74.71**	**82.07**

5 Conclusion and Future Work

We proposed a technique for detecting identity cloning in SocSen services with two components: 1) detecting similar accounts using non-sensitive profile data and a deep forest model, and 2) verifying if similar accounts belong to the same user by requiring the newer account to authenticate with the older account's private key.

Our future work will aim to explore additional datasets as they become available or develop methods to augment our existing data to further validate our approach. We also plan to integrate large language models (LLMs), which could further complement our proposed similar identity detection approach by enhancing the analysis of user-generated content, such as posts and descriptions. In addition, we will investigate how models like XGBoost and LightGBM could complement or enhance our approach, especially in terms of detection accuracy and computational efficiency. We plan to explore advanced techniques for refining weak label generation, such as incorporating confidence-based filtering and active learning strategies, to further enhance the robustness of our model against noisy labels and improve its overall performance. Finally, we target making real-world case studies to evaluate the proposed cryptography-based authentication approach in practice.

References

[1] Advanced encryption standard (AES). NIST, 2001.

[2] T. Aamir, H. Dong, and A. Bouguettaya. Social-sensor composition for tapestry scenes. *IEEE Trans. Serv. Comput.*, pages 1–1, 2020.

[3] Tooba Aamir, Athman Bouguettaya, Hai Dong, Abdelkarim Erradi, and Rachid Hadjidj. Social-sensor cloud service selection. In *Proc. of IEEE ICWS*, pages 508–515, 2017.

[4] Tooba Aamir, Athman Bouguettaya, Hai Dong, Sajib Mistry, and Abdelkarim Erradi. Social-sensor cloud service for scene reconstruction. In *Proc. of ICSOC*, pages 37–52, 2017.

[5] Tooba Aamir, Hai Dong, and Athman Bouguettaya. Social-sensor composition for scene analysis. In *Proc. of ICSOC*, pages 352–362, 2018.

[6] Muhammad Al-Qurishi, Mabrook Al-Rakhami, Atif Alamri, Majed Alrubaian, Sk Md Mizanur Rahman, and M Shamim Hossain. Sybil defense techniques in online social networks: a survey. *IEEE Access*, 5:1200–1219, 2017.

[7] Ahmed Alharbi, Hai Dong, Xun Yi, and Prabath Abeysekara. NPS-AntiClone: Identity cloning detection based on non-privacy-sensitive user profile data. In *Proc. of IEEE ICWS*, pages 618–628, 2021.

[8] Ahmed Alharbi, Hai Dong, Xun Yi, and Prabath Abeysekara. Privacy-aware identity cloning detection based on deep forest. In *Proc. of ICSOC*, pages 415–430. Springer, 2021.

[9] Ahmed Alharbi, Hai Dong, Xun Yi, and Prabath Abeysekara. Cloned identity detection in social-sensor clouds based on incomplete profiles. *IEEE Transactions on Services Computing*, 17(6):3227–3240, 2024.

[10] Ahmed Alharbi, Hai Dong, Xun Yi, Zahir Tari, and Ibrahim Khalil. Social media identity deception detection: A survey. *CSUR*, 54(3):1–35, 2021.

[11] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proc. of WWW*, pages 551–560, 2009.

[12] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Proc. of CRYPTO*, pages 213–229. Springer, 2001.

[13] Leo Breiman. Random forests. *Mach. Learn*, 45(1):5–32, 2001.

[14] J Douglas Carroll. Generalization of canonical correlation analysis to three or more sets of variables. In *Proc. of APA convention*, pages 227–228, 1968.

[15] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer, 2012.

[16] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proc. of IMACC*, pages 360–363. Springer, 2001.

[17] William W Cohen, Pradeep Ravikumar, and Stephen E Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proc. of IIWeb*, pages 73–78, 2003.

[18] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. 1999.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[20] MA Devmane and NK Rana. Detection and prevention of profile cloning in online social networks. In *Proc. of ICRAIE*, pages 1–5, 2014.

[21] Hai Dong, Farookh Khadeer Hussain, and Elizabeth Chang. A survey in traditional information retrieval models. In *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies*, pages 397–402, 2008.

[22] Michael Fire, Roy Goldschmidt, and Yuval Elovici. Online social networks: threats and solutions. *IEEE Communications Surveys & Tutorials*, 16(4):2019–2036, 2014.

[23] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Mach. Learn*, 63(1):3–42, 2006.

[24] Oana Goga, Giridhari Venkatadri, and Krishna P Gummadi. The doppelgänger bot attack: Exploring identity impersonation in online social networks. In *Proc. of IMC*, pages 141–153, 2015.

[25] M Choudary Gorantla, Raju Gangishetti, and Ashutosh Saxena. A survey on id-based cryptographic primitives. *IACR Cryptol. ePrint Arch.*, 2005:94, 2005.

[26] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proc. of SIGKDD*, pages 855–864, 2016.

[27] Aditi Gupta, Hemank Lamba, Ponnurangam Kumaraguru, and Anupam Joshi. Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In *Proc. of WWW*, pages 729–736, 2013.

[28] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. 1992.

[29] Bing Huang, Hai Dong, and Athman Bouguettaya. Conflict detection in iot-based smart homes. In *2021 IEEE International Conference on Web Services (ICWS)*, pages 303–313, 2021.

[30] Gordhan Jethava and Udai Pratap Rao. A novel defense mechanism to protect users from profile cloning attack on online social networks (osns). *Peer-to-Peer Netw. Appl.*, 15(5):2253–2269, 2022.

[31] Shunhui Ji, Shaoqing Zhu, Pengcheng Zhang, Hai Dong, and Jianan Yu. Test-case generation for data flow testing of smart contracts based on improved genetic algorithm. *IEEE Transactions on Reliability*, 72(1):358–371, 2023.

[32] Lei Jin, Hassan Takabi, and James BD Joshi. Towards active detection of identity clone attacks on online social networks. In *Proc. of CODASPY*, pages 27–38, 2011.

[33] Georges A Kamhoua, Niki Pissinou, SS Iyengar, Jonathan Beltran, Charles Kamhoua, Brandon L Hernandez, Laurent Njilla, and Alex Pissinou Makki. Preventing colluding identity clone attacks in online social networks. In *Proc. of ICDCSW*, pages 187–192, 2017.

[34] Georgios Kontaxis, Iasonas Polakis, Sotiris Ioannidis, and Evangelos P Markatos. Detecting social network profile cloning. In *Proc. of PERCOM Workshops*, pages 295–300, 2011.

[35] Faiza Masood, Ahmad Almogren, Assad Abbas, Hasan Ali Khattak, Ikram Ud Din, Mohsen Guizani, and Mansour Zuair. Spammer detection and fake user identification on social networks. *IEEE Access*, 7:68140–68152, 2019.

[36] Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. Twitter under crisis: Can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79, 2010.

[37] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *J. Big Data*, 2(1), 2015.

[38] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proc. of EMNLP-IJCNLP*, 2019.

[39] Alberto Rosi, Marco Mamei, Franco Zambonelli, Simon Dobson, Graeme Stevenson, and Juan Ye. Social sensors and pervasive services: Approaches and perspectives. In *Proc. of PERCOM Workshops*, pages 525–530, 2011.

[40] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Proc. of Eurocrypt*, pages 457–473. Springer, 2005.

[41] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of EUROCRYPT*, pages 47–53. Springer, 1984.

[42] Le Sun, Hai Dong, and Alex X. Liu. Aggregation functions considering criteria interrelationships in fuzzy multi-criteria decision making: State-of-the-art. *IEEE Access*, 6:68104–68136, 2018.

[43] Madhura Vyawahare and Sharvari Govilkar. Fake profile recognition using profanity and gender identification on online social networks. *Soc. Netw. Anal. Min.*, 12(1):170, 2022.

[44] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE transactions on parallel and distributed systems*, 22(5):847–859, 2010.

[45] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[46] Pengcheng Zhang, Huiying Jin, Hai Dong, Wei Song, and Liyan Wang. La-lmrbf: Online and long-term web service qos forecasting. *IEEE Transactions on Services Computing*, 14(6):1809–1823, 2021.

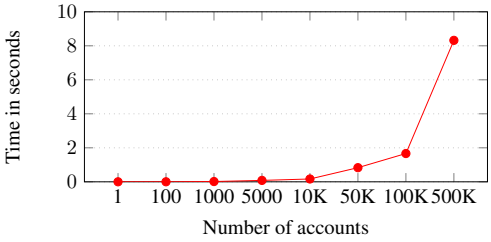
[47] Pengcheng Zhang, Bin Ren, Hai Dong, and Qiyin Dai. Cagfuzz: Coverage-guided adversarial generative fuzzing testing for image-based deep learning systems. *IEEE Transactions on Software Engineering*, 48(11):4630–4646, 2022.

[48] Xianghan Zheng, Zhipeng Zeng, Zheyi Chen, Yuanlong Yu, and Chunming Rong. Detecting spammers on social networks. *Neurocomputing*, 159:27–34, 2015.

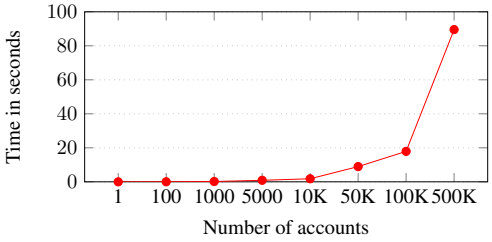
[49] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[50] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *Natl. Sci. Rev.*, 5(1):44–53, 2018.

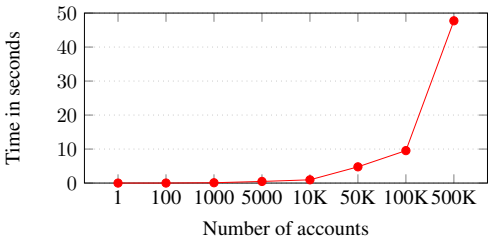
[51] Zhi-Hua Zhou and Ji Feng. Deep forest: towards an alternative to deep neural networks. In *Proc. of IJCAI*, pages 3553–3559, 2017.



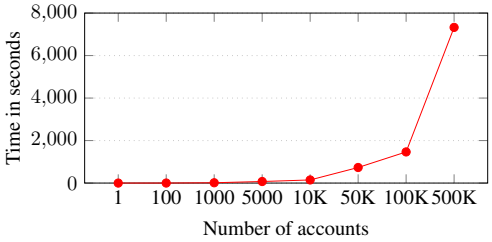
(a) Private key generation Extract(Sec. 3.3.3))



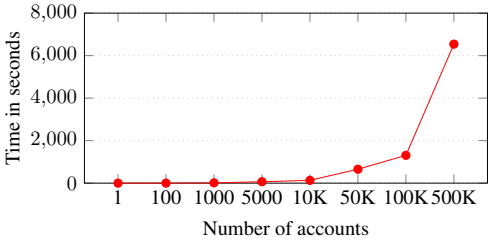
(b) Private key encryption (AES-Enc$_{PK}$(Sec.3.3.3))



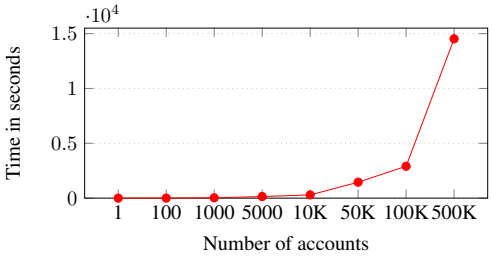
(c) Private key decryption (AES-Dec$_{PK}$ (Sec.3.3.3))



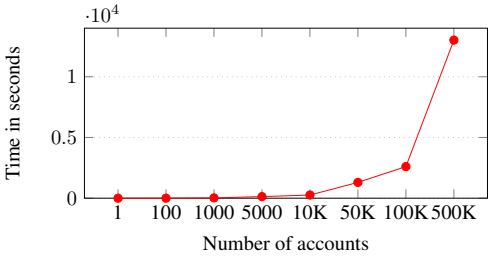
(d) Session key Encryption (Encrypt (Sec.3.3.3))



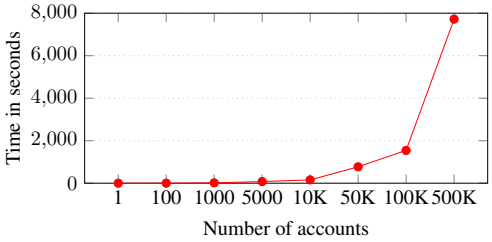
(e) Session key Decryption (Decrypt (Sec.3.3.3))



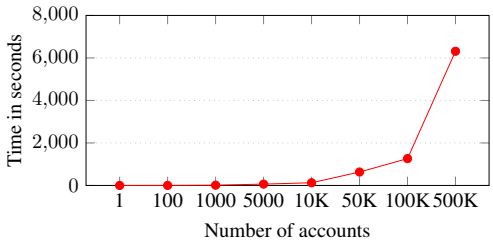
(f) Random message encryption (Encrypt (Sec.3.3.3))



(g) Random message decryption (Decrypt (Sec.3.3.3))

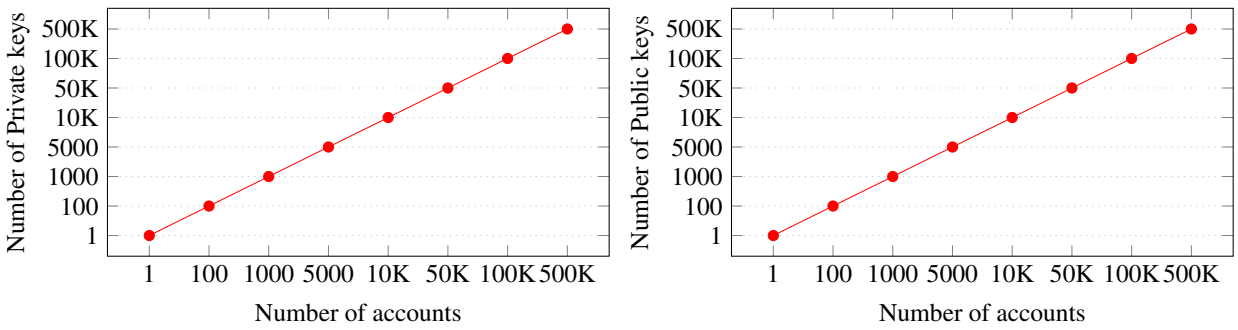


(h) Response encryption (Encrypt (Sec.3.3.3))



(i) Response decryption (Decrypt (Sec.3.3.3))

Figure 11: Time efficiency



(a) Private key storage

(b) Public key storage

Figure 12: Key storage