2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

DIVa: Decentralized Identity Validation for Social Networks

Amira Soliman*, Leila Bahri†, Barbara Carminati†, Elena Ferrari† and Sarunas Girdzijauskas*
*Electrical Engineering School
KTH - Royal Institute of Technology, Sweden
Email: {aaeh, sarunasg}@kth.se
†STRICT SociaLab
Insubria University, Italy
Email: {leila.bahri, barbara.carminati, elena.ferrari}@uninsubria.it

***Abstract*—Online Social Networks exploit a lightweight process to identify their users so as to facilitate their fast adoption. However, such convenience comes at the price of making legitimate users subject to different threats created by fake accounts. Therefore, there is a crucial need to empower users with tools helping them in assigning a level of trust to whomever they interact with. To cope with this issue, in this paper we introduce a novel model, *DIVa*, that leverages on mining techniques to find correlations among user profile attributes. These correlations are discovered not from user population as a whole, but from individual communities, where the correlations are more pronounced. DIVa exploits a decentralized learning approach and ensures privacy preservation as each node in the OSN independently processes its local data and is required to know only its direct neighbors. Extensive experiments using real-world OSN datasets show that DIVa is able to extract fine-grained community-aware correlations among profile attributes with average improvements up to 50% than the global approach.**

Keywords. Community-aware Identity Validation, Ensemble Learning, Privacy-preserving Learning, Decentralized Online Social Networks.

I. Introduction

Online Social Networks (OSNs), such as Facebook, Twitter, or LinkedIn, have attracted millions of users and have put online virtual interactions at the same level of importance of offline ones. However, current OSNs still lag behind in providing completely safe and secure online services and are subject to a variety of security threats, such as spam, malware, and phishing attacks [1], [2], [3]. For example, Facebook recently announced that the portion of fake and duplicate accounts is between 5.5% and 11.2% [4].

Furthermore, OSNs incentivize users to share their news in their public space, yet users do not have a clear idea of who accesses their personal information. In addition to users' friends, these pieces of information can be accessible by third-party applications, data aggregators, or external applications [5]. Therefore, several research initiatives have tried to increase users' awareness towards security breaches in OSNs, and some research and open-source communities have implemented and deployed Decentralized Online Social Networks (DOSNs) operating without a centralized provider, such as Diaspora,[1] and Jappix[2]. The main objectives behind decentralization are preserving users privacy in both shared content and communication, and the complete freedom from any form of censorship, tracking, or profiling.

However, the lightweight process for obtaining identities to join either OSNs or DOSNs (e.g., confirming a valid email address) underpins the vulnerability of such networks to undergo different attacks. Moreover, users are given the complete freedom to fill up the records of their profiles without validating them. Thus, malicious users can provide misleading information or they can easily claim to be someone else [6], [7], making the networking environment unsafe. Several researchers have addressed this issue either though techniques for fake accounts detection [8], [9], [10], or from an identity management and validation perspective. For instance, [11] suggests to evaluate an identity on a given network based on feedback of its connections on another one. Cai et al. [12] suggest people to people recommendations by relying on collaborative filtering. In [13], users are suggested to be identified from their typing patterns; whereas chatting patterns are exploited in [14]. More recently, [15] suggests identifying users across networks based on geo-location and time-stamp information of their posts. All these pieces of work rely on using users' sensitive data that might hinder their privacy. Moreover, there is no framework for users to evaluate the perceived trustworthiness of their new online contacts.

Into a step towards the automation of identity validation, few works have addressed the issue using profile information only. For example, by introducing game theoretical models, Squicciarini et al. [6] describe providers/users interaction as a two-players game by which identities are validated based on the assessment of trade-offs between benefits against risks of allowing them to join the network. Sirivianos et al. [11] propose the same model with the validation part being performed via feedback of new user's direct friends regarding the trustworthiness of provided information.

More recently, [16] suggests validating profiles based on correlations between their attributes. The authors demonstrate that existing correlations between profile attributes can be reliably used to estimate a profile's identity trustworthiness from its attribute values only. For example, a user who specifies

[1] https://www.joindiaspora.com/
[2] https://jappix.com/

ASONAM '15, August 25-28, 2015, Paris, France
 ISBN 978-1-4503-3854-7/15/08 $15.00
DOI: http://dx.doi.org/10.1145/2808797.2808861

her occupation as a *computer science student* is expected to be attending a *university* that offers such major. More precisely, the authors suggest a two phase system. In the first one, they exploit a supervised crowd-based learning strategy to extract profile attribute correlations that are meaningful from an identity validation perspective. They do this by gathering human feedback from a group of trusted users on a centralized profiles training dataset. Once these correlations are identified, they are used in the second phase, that also engages users' feedback, to estimate the identity trustworthiness of a target profile.

Although profile exploitation approach is very promising, there are several challenges that hinder its success in terms of practicality. First, and given the number of users in current OSNs, it is *not realistically scalable* to rely on trusted users feedback for the learning of attribute correlations. Besides, collecting the data centrally violates DOSNs *privacy* constraints as it is often not allowed to move users' data outside their direct connections. Furthermore, it is hard to identify who the trusted users are to ensure the accuracy of learning attribute correlations. Finally, relying on users' feedback introduces *privacy* risks and is not applicable for DOSN models.

Additionally, decoupling profiles data from the semantic of users' connectivity during attribute correlations extraction degrade the *performance* as it might result in infirm and/or prejudiced validation. Specifically, social networks exhibit a clustering phenomena by which users topologically cluster into communities [17], [18]. Therefore, basing the learning on all the profiles as a homogeneous entity might capture generalizations that might be very generic, or on the contrary that might only be expressed in some communities. Thus, some users will be penalized for mismatching some validations that could be unsuitable for the communities they want to join. For example, occupation in scientific domains require specific majors while it is not always the case in artistic fields. Universal learning will enforce the correlation between job and major, if the majority of users have scientific backgrounds, thus smaller communities of artistic users will be penalized for mismatching this rule.

To address these issues, in this paper we propose a Decentralized Identity Validation framework (DIVa). The key of DIVa's performance lies in its ability to leverage on homogeneity among users' profiles inside every existing community instead of human-feedback based learning. DIVa employs recently developed high performance decentralized diffusion-based community detection strategy [19]. This allows DIVa to extract more meaningful profile attribute correlations (i.e., correlated attribute sets - CAS) within communities than any state-of-the art method. In addition, all DIVa's components are running node-centric algorithms (designed to never use global knowledge), making it scalable and be the first framework, to our knowledge, suitable for DOSNs. Although DIVa was designed for decentralized environments, due to the nature of its underlying node-centric algorithms it also excels in centralized settings too, in particular on current graph processing frameworks like Graphlab [20].

Scalability: DIVa extends the ensemble learning paradigm in distributed machine learning and works on fully distributed datasets without collecting the data into one central location. DIVa's structure is a network of nodes such that each executes a DIVa instance. DIVa instances use their local data and generate a set of distributed models by exploiting principles of Association Rule Mining (ARM) [21]. Before being exchanged with neighboring nodes, such models are updated against the local data, advancing the learning process without the need of actual data leaving the home-node. Next, communities CASes are computed in an incremental fashion by aggregating these intermediate models per community. This achieves the same predictive and analytic power in distributed fashion without violating users' privacy.

Privacy: to preserve users' privacy, instances of DIVa have access only and solely to the local data, that is available given particular privacy settings. DIVa instances extract their local correlations using profiles of their direct neighbors, then exchange only the set of generated models within their communities to agree on the community CAS. This preserves users' privacy as data is processed locally. Figures 1 and 2 illustrate the three phases of DIVa and interactions among DIVa instances.

Performance: the key of DIVa's performance relies on successfully identifying meaningful communities, hence the community detection algorithm that we harness goes beyond the current state-of-the art. Majority of research in community detection focuses on partitioning social network into disjoint components. However, in social networks, every individual typically belongs to more than one community, such as the community of family members (e.g., that of friends and classmates, that of co-workers, etc). Furthermore, the number of communities a user can belong to is unlimited as a person can simultaneously associate with as many groups as he wishes. Thus, for high quality results it is imperative that we perform overlapping community detection. Therefore, DIVa allows users to have multiple community memberships. Moreover, the used community detection algorithm works in a massively parallel manner as well as requires only the local knowledge of nodes' direct neighbors. Further, DIVa detects topological changes of previously detected communities caused by newly joining nodes and reforms the new communities and recomputes their CASes. Therefore, DIVa can be used as online framework that exploits incremental communities changes.

The main contributions of this work are:

- a *scalable* and massively parallel identity validation model that suitably fits DOSNs and OSNs environments,
- fully automated *privacy preserving* identity validation scheme,
- *community-aware validation* model based on extracting association rules that reveals mostly frequent fine-grained identity patterns inside each community,
- identity validation model that allows users to have *multiple community memberships*, and
- *incremental operation* of community-aware validation by continually updating validation rules upon newly joining users.

We have performed several experiments, using real profile data from Fecebook and Google+, to show the effectiveness of our proposed identity validation model. The results show that

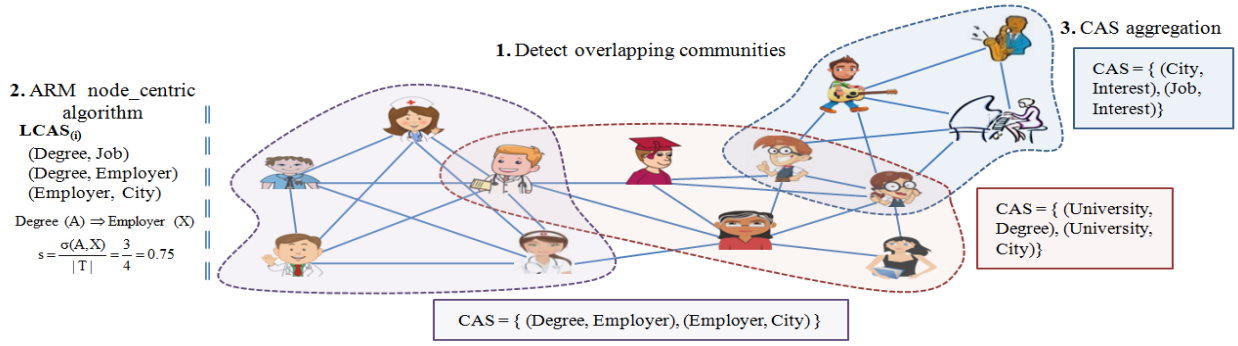

Fig. 1. The three phases of DIVa model. First, communities boundaries are identified by detecting densely connected regions, then every node performs local learning, and the final step is aggregate rules per community. Meanwhile, universal learning generates the attribute pairs: (Degree, Employer), (Degree, University) that are not expressed in all of existing communities.

DIVa extracts more validation rules than the global approach. Most importantly, DIVa provides more effective validation using fine-grained and community-aware correlations with average improvements of up to 50% in Facebook and 16% in Google+ than the general correlations extracted by universal approach.

The rest of the paper is organized as follows. Section II defines the DIVa model, Section III provides the security analysis, whereas Section IV presents and discusses experiments results. Finally, Section V concludes the paper and presents planned future work. For space limitation, further technical details on DIVa algorithms (Section II), proofs to security theorems (Section III), and detailed complexity analysis (Section IV.D) are presented in a technical report [22].

II. DIVa: Decentralized Identity Validation

We design DIVa based on the principle that OSNs exhibit a clustering phenomena as users topologically cluster into groups with intra-ties denser than inter-ties [17], [18]. Besides, users in a group typically have high similarity to each other sharing common identity and background trends [18]. From this, we derive two characteristics of OSNs that we exploit in our model: (1) OSNs cluster into socially homogeneous communities that, (2) bind people with generally common identity trends and patterns. Accordingly, DIVa operates in three phases (see Figure I) to generate community specific CASes that can be used to evaluate the integrity of profile information of new nodes desiring to join a community. However, the focus of this work is to learn the CASes that can be then used in different ways for validation purposes. Example 1 exemplifies a possible scenario for using DIVa for identity validation.

Example 1. Assume Mike is an OSN user belonging to community C_m only. Assume that DIVa finds that {Education, Job}, and {Job, Current City} are CASes in C_m. Given a new user Alice wants to connect with Mike, he can estimate her profile's trustworthiness by checking the values on her profile corresponding to his communities' CASes. To better assist Mike, DIVa can also provide him with the top-n values associated with each CAS in C_m.

We provide the details of each of DIVa's three phases in the following subsections.

A. Discovering Local CASes

We base the distributed learning of the LCASes on the association rule mining (ARM) that is generally used for the extraction of associations among different items in a shopping basket [21]. In our model, the items are the profile attributes and the association rules are the correlated attribute sets. For example, a node can learn that among her direct friends, users who are employed at company X also live in city Y. If a node observes that this is frequent enough in the profiles of its direct friends, the node deduces that attributes Employer and City are correlated. Before giving the formal definitions, we first introduce some notations.

We model an OSN as an undirected graph G = (V, E), where V is the set of nodes (or users) and E is the set of edges (or friendships), where $e_{ij} \in E$ denotes a relationship between nodes v_i and $v_j \in V$. We denote with S = $\{A_1, A_2, ..., A_m\}$, the profile schema adopted in the OSN. Given a node $v_i \in V$, p_i denotes the set of its profile values: $p_i = \{p_i.a_1, p_i.a_2, ..p_i.a_m\}$, where $p_i.a_k$ is the value provided by v_i for $A_k \in S$.

In the decentralized learning of LCAS, every node stores the profile information of all its direct friends to form its *Local Profile Collection*. More precisely, given $v_i \in V$, we introduce the set $DF_i = \{v_j \in V | e_{ij} \in E\}$ as the set of v_i's direct friends, and $LPC_i = \{p_k | v_k \in DF_i\}$ as the collection of their profiles, referred to as v_i's local profile collection.

To minimize computational effort, a node v_i considers only those attributes having high value frequency in its LPC_i. As an example, a job value *musician* that is repeated in more than 30% of the profiles in LPC_i makes the attribute *Job* a frequent one; whereas, one satisfying less than this threshold is a non-frequent attribute. Therefore, we define:

Definition 2.1: **Local Frequent Attributes - LFA**. Let $v_i \in V$ and LPC_i be its local profile collection. Let $A_k \in S$ be an attribute from the profile schema and let $P_k^{\vartheta} \subseteq LPC_i$ be the set of profiles in LPC_i having the same value ϑ for attribute A_k. That is, $\forall\ \vartheta$ (ϑ is a given value), $P_k^{\vartheta} = \{p_m \in P_k^{\vartheta} | p_m.a_k = \vartheta\}$. Let $LFA_i \subseteq S$ be the set of attributes such that, $LFA_i = \{A_k \in LFA_i | \exists \vartheta$ s.t. $\frac{|P_k^{\vartheta}|}{|LPC_i|} \geq \epsilon\}$, where ϵ is a global predefined threshold.

Given LFA_i, v_i computes the support of each attributes pair in it. We limit to a pair as any combination larger than 2 can be decomposed into correlations between pairs.[3]

We define the support of an attributes pair as follows:

Definition 2.2: **Support of an attributes pair**. Let $v_i \in V$, LPC_i be its local profile collection, and LFA_i be its local frequent attributes set. Let (A_j, A_h) be a pair of attributes from LFA_i. The support of (A_j, A_h) defines the percentage of co-occurrence of the same paired values for the two attributes A_j and A_h to the total number of values in LPC_i:

$$Support((A_j, A_h)) = \frac{\text{values-co-occurrence}(A_j,\ A_h)}{\text{all-values}(A_j,\ A_h,\ LPC_i)} \quad (1)$$

Where,

values-co-occurrence(A_j, A_h)=$|\{(p_e, p_m) \in LPC_i | p_e.a_j = p_m.a_j \wedge p_e.a_h = p_m.a_h\}|$.

and,

all-values(A_j, A_h, LPC_i)=$|\{\vartheta \ | \exists p \in LPC_i$ s.t., $p.a_j = \vartheta \vee p.a_h = \vartheta\}|$

By Equation 1, the support of an attributes pair is computed based on the values-co-occurrence of its elements to the total number of values in a local profile collection. For example, the values-co-occurrence between *Job* and *Education* is the number of profiles in which a specific pair of values, (Job = X, Education = Y), exist. However, the values of profile attributes in an OSN are mostly textual and are input as free-text by the users. Thus, two values might be syntactically different or worded differently but semantically the same. Therefore, computing the values-co-occurrence requires a technique for values comparison. We use for that a set of intersecting words comparison by which we compute the set of common words between two attributes' values. The size of this set reflects the similarity between these two values (see Algorithm 1).

Once the support has been calculated for all pairs in LFA_i, node v_i selects the ones for which the support is high enough to reflect they are correlated. Formally, we define a LCAS as:

Definition 2.3: **Local Correlated Attribute Set - LCAS**. Let $v_i \in V$ and $LFA_i \subseteq S$ be its local frequent attributes set. Let (A_j, A_h) be a pair of attributes from LFA_i. The pair (A_j, A_h) is a local correlated attribute set, denoted as LCAS, if: Support((A_j, A_h)) $\geq \beta$, where β is a global predefined threshold.

Algorithm 1 details how node v_i finds its LCAS list (i.e., $CLIST_i$). $CLIST_i$ is a list of attribute pairs (i.e., (A_1, A_2)) with their support contained in the variable $(A_1, A_2).c$. First, for every (A_1, A_2) from LFA_i, the pair is inserted in $CLIST_i$ with a support equal to 0 (line 3). Then, *tokenize()* retrieves all the words from all the values of A_1 and A_2 in all the profiles in LPC_i (lines 4 and 5) to form their respective word list, W_1 and W_2. Then, we calculate the number of pairs of profiles from LPC_i, such that their values for A_1 and A_2 share some words from their respective word lists and we update $(A_1, A_2).c$ accordingly (*getIndex()* locates the pair (A_1, A_2) in $CLIST_i$) (lines 6 - 12). Basically, for every two profiles from LPC_i, we get the set of words for which word lists of A_1 and A_2 intersect, denoted as X and Y respectively. Then, the normalized support of (A_1, A_2) is computed by dividing the length of minimum intersection between their word lists by all the words in W_1 and W_2 (line 10). The pairs in $CLIST_i$ with support higher than the predefined threshold β are the LCASes of v_i.

Example 2. Consider Example 1 and assume Jane as another OSN user belonging to communities C_m and C_j. To learn her LCAS, Jane collects the available profile attributes from all her direct friends to construct LPC_{Jane}. Assume she finds that *Job* and *City* are in LFA_{Jane}; that is, their values are highly frequent in LPC_{Jane}. Jane computes the number of profile pairs in LPC_{Jane} for which these two attributes' pair have similar values. Assume in more than 40% of the profiles in LPC_{Jane}, this pair is co-occurring. Assuming that the LCAS threshold β = 0.3, the pair *(Job, City)* is an LCAS for Jane.

Algorithm 1 LCAS Learning at node v_i

Require: LFA_i, LPC_i, and β
Ensure: List of LCAS with their support: $CLIST_i$
1: $CLIST_i \leftarrow \emptyset$
2: **for all** $(A_1, A_2) \in LFA_i$ **do**
3: $CLIST_i \leftarrow$ insert((A_1, A_2), 0)
4: $W_1 \leftarrow$ tokenize(LPC_i, A_1)
5: $W_2 \leftarrow$ tokenize(LPC_i, A_2)
6: **for all** $p_k, p_j \in LPC_i$ **do**
7: $X \leftarrow$tokenize($p_k.a_1$) $\cap$ tokenize($p_j.a_1$)
8: $Y \leftarrow$tokenize($p_k.a_2$) $\cap$ tokenize($p_j.a_2$)
9: **if** $X \neq \varnothing \wedge Y \neq \varnothing$ **then**
10: $s \leftarrow \frac{min(|X|,|Y|)}{|W_1|+|W_2|}$
11: $CLIST_i[getIndex(A_1, A_2)].c \leftarrow CLIST_i[getIndex(A_1, A_2)].c + s$
12: **end if**
13: **end for**
14: **end for**
15: **for all** $(A_1, A_2) \in CLIST_i$ **do**
16: **if** $CLIST_i[getIndex(A_1, A_2)].c < \beta$ **then**
17: remove($CLIST_i$, (A_1, A_2))
18: **end if**
19: **end for**

B. Decentralized Community Detection

LCASes learned locally at nodes need to be disseminated to construct community level CASes. Thus, a community detection step is required. In general, a community is defined as a subgraph G' $\subset$ G = (V', E') representing a tightly-knit set of nodes that are sparsely connected to the rest of the nodes in G.

Many decentralized algorithms for community detection have been proposed [23], [24], [19]. In particular, the work by Rahimian et al. [19] seems to comply with our decentralization requirement. In this work, every node starts as a community by itself using its node ID as its community ID. Then, every node chooses to quit its current community and join one of its neighbour's if this brings some modularity gains. Therefore, we define modularity gain in terms of dominant ID by which every node changes its community ID to the dominant one

[3]Assume CAS = (A,B,C). This implies there is a non-empty set of profiles where the values for A, B, and C co-occur. That is (A, B), (B, C), and (C, A) are correlated. The correlations (A, B), (B, C), and (C, A) are not enough for (A, B, C) to be correlated; however, the correlation (A, B, C) necessarily implicates the paired correlations.

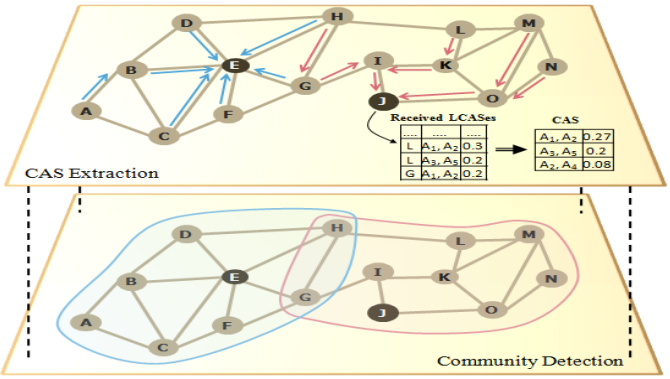

Fig. 2. Identifying two overlapping communities and communicating with diva nodes to perform community-level aggregation.

in its neighbours. This step is iterated until no node wants to change its ID as it already represents the dominant one of all its neighbors. If a node does not find a dominant ID among its neighbours, it changes to the highest ID between its own and the ones of its neighbours.

However, since communities in social networks are majorly overlapping, and considering that DIVa aims to learn correlations that are more representative of a node's environment, we opt for a soft clustering approach. That is, a node can belong to more than one community and hence have more than one dominant community ID. More precisely, a node keeps track of the top dominant community IDs that it learns about when the community detection algorithm converges. As shown in Figure 2, DIVa instances have organized themselves into two overlapping communities. Further, node with the largest ID (i.e., node with darker shade) has been identified as the leader node of the detected community.

C. Community-level Aggregation

After step 2 of DIVa, every node becomes aware of its community ID(s). Furthermore, each community ID points to its leader node (i.e., its diva[4] node) of the community; that is the node with the maximum nodeID. Particularly, this node is responsible for collecting all LCASes found in its community to extract from them the community CASes using a weighted voting mechanism. Thus, each node sends its LCASes and their corresponding support values to its community's diva(s) and receives back CASes with the maximum votes represented by maximum aggregated support. However, recall that in DOSNs nodes are aware only of their direct friends. Therefore, nodes must be aware of the path to reach their communitie(s)' diva(s) by a hop-by-hop routing over their social ties. Accordingly, during the execution of the community detection part, nodes maintain paths leading to the diva of each community they belong to. Path construction is straightforward. First, a node checks if the diva is a direct friend. If it is not, the node creates the path by assembling the node IDs of intermediate nodes leading to the diva. Figure 2 depicts how nodes reach their communities diva nodes by following the constructed path towards them.

Given that all nodes know the path to each of their communitie(s)' diva node(s), messages containing their LCAS, along with the support of every LCAS attached as a parameter $LCAS.c$, are sent by every node to reach the destined diva(s) using a hop-by-hop consumption of the specified path. With every newly received message, each diva node inserts the ID of the sender into a $participants$ list and the communicated LCAS into a list of received LCAS (i.e., $LIST_L$). A diva node is assumed to be preliminary aware of the size of its community after execution of community detection. When the number of received participants represents the majority of the community's nodes, the diva aggregates the support of received LCAS. The calculation of the aggregated support is straightforward: for each LCAS a diva node receives, it computes the average of its received supports from the different nodes communicating it. As shown in Figure 2, node *J* has received LCASes form other nodes belonging to the community and has aggregated partial results to reach community CAS. The last step is to inform the rest of nodes with the final consensus in CAS representing the community.

Example 3. Following up with Example 2, Jane communicates to the diva nodes of C_j and C_m her LCAS $(Job, City)$ with its computed support. Assuming that the diva of C_j (i.e., $diva_j$) receives this LCAS from the majority of nodes in C_j, $(Job, City)$ is a CAS in C_j. However, let us assume that in C_m, $(Job, City)$ is under represented and hence it is not a CAS in C_m. Given this, Jane will have the choice of using the CAS $(Job, City)$ to evaluate the trustworthiness of her new contacts based on which community, C_j or C_m, she knows them from. For example, a new contact who claims to be a colleague is more highly expected to meet this CAS, when another one who Jane knows from the GYM is less likely to be judged based on it.

III. Security Analysis

We analyze the security of DIVa assuming a malicious adversary model. The goal of an attacker would be to convince the community to use a corrupted CAS so as to confirm the trustworthiness of his/her identity or to compromise the ones of honest nodes. Since CAS values depend on the co-occurrence frequency of similar attribute values within a community, an attacker can have an effect only by introducing enough fake profiles (i.e., sybil nodes [25]) within the target community. Sybil nodes in OSNs have been studied both from a graph and a profile perspectives [26]. Since we detect communities based on topology, our focus will be on graph based sybil detection (GSD). Most work on GSD agrees that Sybils often manifest in proper topological structures making them distinguishable from honest nodes [26]. More precisely, sybil nodes exhibit proportionally smaller degree centrality and tend to group as outliers in the graph, compared to honest nodes [26]. Therefore, based on the logic of DIVa, sybil nodes would be detected in separate communities with their own CASes unless they trick enough honest users into establishing friendship links with them so that they join their honest communities. This last possibility is discussed in the following sub-sections. Proofs to security theorems are available in [22].

A. Introducing a fake CAS

To introduce a new CAS, CAS_{new}, to a community, an attacker has to successfully integrate in it enough fake nodes, say z nodes, that carry profile information confirming

[4]Diva is the main female singer in an opera company. In analogy, we name community leader nodes as diva nodes.

CAS_{new}. Inserting a node x into a community C requires the creation of a number of edges (i.e., friendships) with enough other members of C so that x is included into it by the community detection algorithm. We emphasize on the amount of effort needed to introduce one new fake node into a community and we deterministically define z, the number of such fake nodes required to introduce a fake CAS_{new}.

Theorem 3.1: Let $\overline{C} \subset G$, $\overline{C} = (\overline{C}.V, \overline{C}.E)$, be a community of size n ($|\overline{C}.V| = n$). Let sup_{lowest} be the lowest support by which a CAS is accepted in $\overline{C}$. For a new CAS, CAS_{new} to appear in $\overline{C}$, it must be inserted a group of fake nodes C_f that successfully join $\overline{C}$ and that show profile information confirming CAS_{new} such that:

$$z = |C_f| \geq \frac{sup_{lowest}}{(1-sup_{lowest})} * n.$$

By Theorem 3.1, we can see that the required adversary effort to introduce a fake CAS is directly proportional to the size of the target community and to the lowest support by which it accepts a new CAS. Given that, communities tend to be more vulnerable to adversaries when they are small in size. However, for the adversary to succeed, fake nodes need to successfully establish enough links with the members of the target community. We believe that small communities, that are supposedly composed of nodes knowing and trusting each other, would tend to be more selective and hence more resilient to accepting unknowns compared to larger communities. In addition to that, a smaller community is expected to be more homogeneous (a small group of people knowing each other closely would be more homogeneous than a larger group) and so could allow high threshold values for the computations of its CASes. By Theorem 3.1, it is clear that the higher the support threshold is, the higher the number of fake nodes are needed for an attack to succeed.

B. Corrupting a valid CAS

Additionally, an adversary might be interested in removing a valid CAS from a community. This can be achieved by introducing profiles that are not compliant with this CAS in a number big enough to disturb its valid support.

Theorem 3.2: Let $\overline{C} \subset G$, $\overline{C} = (\overline{C}.V, \overline{C}.E)$, be a community of size n ($|\overline{C}.V| = n$). Let sup_{lowest} be the lowest support by which a CAS is accepted in $\overline{C}$. For a valid CAS, CAS_{valid} with support S_v, to disappear from $\overline{C}$, it must be inserted in $\overline{C}$ a group of fake nodes, C_f, that does not have profile information confirming CAS_{valid} such that:

$$z = |C_f| > \frac{S_v * n}{sup_{lowest}} - n.$$

By Theorem 3.2, the vulnerability of a valid CAS is inversely proportional to the percentage of nodes in the community that contributed in making it arise. In fact, the lower the support of a CAS is, the smaller the number of nodes in the community represent it, and as such, the most vulnerable CAS to such an attack is the one having the lowest support. However, the success of such an attack, even for the weakest CAS, still requires the insertion of a considerable number of Sybil nodes in the target community. As we discussed before, this in itself remains a challenge for attackers.

TABLE I. REAL OSN DATASETS USED IN EXPERIMENTS.

Dataset	Nodes	Edges
Facebook	23,332	28,972
GpJUL	2,417,014	25,016,154
GpAUG	4,349,414	35,544,682
GpSEP	4,388,907	43,060,890

IV. EXPERIMENTAL RESULTS

We have implemented DIVa using GraphLab [20] with two different distributed execution modules. The first module executes our adopted community detection algorithm until it converges so that

every node knows the divas of its communities and the paths toward them. Thereafter, the control is moved to the second module that extracts CASes for every detected community. We compare DIVa discovered CASes with global CASes generated by learning from all profiles in one central repository. That is, global CASes are obtained by executing DIVa's LCAS learning algorithm, but over all the available profiles in the OSN all at once.

As aforementioned, the LCAS learning considers a node's LFA to prune the attributes passing the required values-frequency threshold ϵ (Definition 2.2). We set $\epsilon = 0.2$, as we believe that a pattern supported by more than 20% of the population is statistically meaningful. A further discussion on specifying the threshold value is presented in [22].

A. Datasets Description

We conducted several experiments to validate the effectiveness of DIVa using real profile datasets from Facebook and Google+ (shown in Table 1). We used the Facebook dataset collected and used in [27], and the Google+ dataset publicly available from [28]. The profile schema in the Facebook dataset contains: First Name, Gender, Home County, Education, Job, Current Country, and Interests. Meanwhile, the profile in Google+ datasets has fewer attributes, specifically Occupation, Employment, Education, and Places Lived. The Google+ dataset represents three crawled parts of the OSN collected on July, August, and September in 2011. We denote them as GpJUL, GpAUG, and GpSEP, respectively. Given the fact that the Facebook dataset does not have timestamp on links creation, we test the incremental update of DIVa using the Google+ datatset that comes with this information.

B. Batch Execution

The Facebook dataset was collected using a Facebook app that gathered the friendship links and profiles of the users who launched it. As a result, the collected data makes a graph of connected hubs with no overlapping communities. Therefore, every node belongs to a single community, and communities divas are reached directly without any message routing overhead. The community detection algorithm converged after three rounds detecting 64 communities.

Table 2 lists CASes extracted by DIVa and their equivalent support values for two communities comparing them to the global extracted CASes. For the two communities we list the top five attribute pairs in terms of support value. In fact, DIVa extracts 15 attribute pairs on average, but we show the top 5 due to space limitation. As a first observation, CASes

TABLE II. DIVA EXTRACTED CAS VS. GLOBAL CAS FOR FACEBOOK DATASET.

DIVa CAS Community1		DIVa CAS Community2		Global CAS	
Attribute Pair	**Sup.**	**Attribute Pair**	**Sup.**	**Attribute Pair**	**Sup.**
1:(gender, f.name)	0.14	1:(education, employer)	0.168	1:(ob, interest)	0.335
2:(f.name, h.country)	0.105	2:(employer, interest)	0.164	2:(gender, interest)	0.179
3:(education, job)	0.103	3:(job, h.country)	0.108	3:(education, interest)	0.138
4:(job, employer)	0.102	4:(gender, f.name)	0.105	4:(job, h.country)	0.137
5:(education, interest)	0.085	5:(f.name, h.country)	0.104	5:(education, job)	0.126

extracted by DIVa differ between the two communities. This proves that communities have different identity patterns and that DIVa succeeds in revealing them. This is further confirmed by the common CASes in the two communities as their support values hugely differ in each one of them. For example, the pair (education, interest) has different support values 0.085, and 0.168 in Community1, and Community2, respectively. This emphasizes that the importance of an attribute pair differs from a community to another depending on its social and identity characteristics.

Our second observation is related to the nature of the CASes discovered by DIVa compared to the ones retrieved from global learning, especially with regard to the type (single or multi-value) of attributes they contain. Indeed, the pairs extracted universally capture the common global trends and mainly consist of multi-value attributes. Meanwhile, most DIVa CASes are made of single-value attributes. This is an important factor to consider and a real strength of DIVa as single-value attributes cover more specific identity dimensions that could not be captured from the global learning. Indeed, this latter failed to extract fine-grained and community-aware rules that disappear when considering the whole network as one giant component.

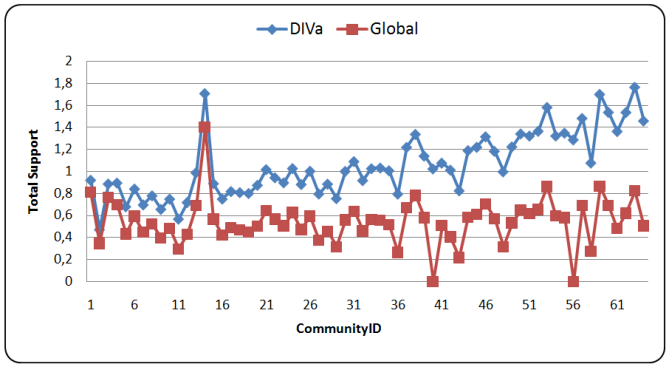

Fig. 3. Comparing total support of DIVa CAS to globally extracted CAS for Facebook dataset.

Figure 3 depicts a comparison of total support values of CASes extracted by DIVa and those extracted globally. For every detected community we compute DIVa total support by summing up the support of its extracted CASes. Meanwhile, the second total support (global) is the sum of the global support values associated with attribute pairs of global CASes if those attribute pairs are correlated inside the detected communities. Figure 3 shows that DIVa provides stronger validation than the global approach, as the total support of DIVa generated CASes is higher than the total support of global CASes. In general, DIVa achieves average improvement over global CASes that is up to 52.5% in the Facebook dataset.

TABLE III. COMMUNITY DETECTION RESULTS FOR GOOGLE+ DATASET.

Dataset	Avg. Comm.	Mod.(H)	Mod.(L)
GpJUL	2.24	0.747	0.372
GpAUG	2.67	0.721	0.421
GpSEP	2.76	0.738	0.492

C. Incremental Execution

Each of Google+ datasets contains timeID with values 0, 1, or 2, indicating which snapshot a directed link between two users appeared in. Thus, we execute our experiments incrementally to update the social graph by adding edges among nodes using timeIDs. Therefore, previously existing nodes monitor topological changes that affect their community membership, and re-execute the community detection module followed by DIVa extraction module when required.

1) Overlapping Community Detection: As aforementioned, community detection in DIVa allows users to have multiple community memberships. This is achieved by having every node updates an ordered list of diva nodes of the dominant communities to which its friends belong throughout the rounds of the algorithm. Specifically, this list is ordered by the number of friends in every community in order to rank community memberships from densely to sparsely connected ones. Next, nodes set their main community ID to diva node of the dominant community among their neighbors. The average number of rounds required for convergence across all Googel+ datasets is 28. Besides, the average path length towards the diva nodes is 5.9, 4.9, and 4.5 in GpJUL, GpAUG, and GpSEP, respectively. This results show that the more connected the graph is, the shorter the path towards the diva nodes is.

Moreover, the results show that on average 95% of all communities overlap with at least one other community, and only 21% of nodes belong to only one community. Table 3 lists the obtained results, for example, the average number of community memberships (Avg. Comm.) in GpJUL dataset is 2.24. Furthermore, the overall average modularity (Mod. (H)) computed for the dominant communities (i.e., the communities with highest rank) for the whole graph is 0.747. The last column in the table lists the overall average modularity (Mod. (L)) computed for the lowest rank communities.

2) Incrementally Updating DIVa CASes: In this set of experiments, we study the effect of the newly added nodes and edges in the social structure of previously detected communities requiring the re-computation of DIVa CASes. Figure 4 shows the percentage of nodes re-performing CASes extraction due to topological changes in their communities. In particular, the lower bound of change should be the percentage of new nodes, where only those nodes execute the community detection module and notify their communities divas with their LCASes. Meanwhile, the upper bound would mean that the

process will start all over from the beginning such that all nodes execute the community detection and LCASes extraction modules. The vertical error bars in Figure 4 represent the range of expected change in the datasets graphs after adding the new nodes.

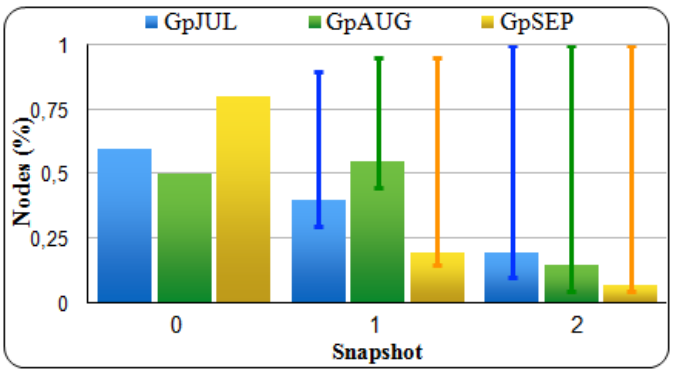

Fig. 4. Percentage of nodes computing DIVa CASes at each snapshot of Google+ datasets.

Intuitively, in the first snapshot all nodes execute both modules. As shown in Figure 4, 60%, 50%, and 80% of GpJUL, GpAUG, and GpSEP, respectively are loaded in the beginning. In the second snapshot 30%, 45%, and 15% new nodes were added to GpJUL, GpAUG, and GpSEP graphs, respectively. The results of the three datasets show that, on average, 17% of old nodes got affected by topological changes caused by the new joining nodes and performed community detection followed by LCASes extraction. The average change reported for adding the last snapshot across all three datasets is only 6%. Consequently, in our framework nodes are able to detect the topological changes surrounding them. Moreover, the results show that these changes are localized and require re-computations only for changed regions not the whole graph.

3) Improvements achieved by DIVa CASes: The results show that DIVa extracts more refined CASes than the global CASes for Google+ datasets. The followings are all DIVa generated CASes: (employer, places_lived) , (major, employer), and (school, major). Moreover, some communities have more CASes such as (school, employer}, (major, places_lived), and (school, places_lived). Meanwhile, the global learning extracted the following four CASes only for the three datasets: (major, employer), (major, places_lived), (school, places_lived), and (school, major).

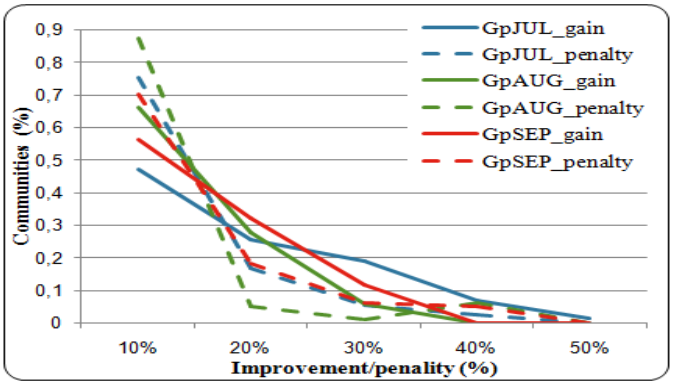

Fig. 5. Comparing DIVa and global CASes, showing DIVa improvement and penalty of applying global CASes.

Figure 5 depicts DIVa achieved improvement compared to global results for all datasets. We compute the improvement by the difference between total support of DIVa CASes and global CASes. On average, for 74% of the communities, DIVa CASes are more in number than the globally extracted CASes. Furthermore, the achieved gain ratio for these communities range from 10% to 50% improvement. The Google+ results emphasize that global CASes capture only the attribute pairs that reflect the global patterns. On the other hand, DIVa succeeds in revealing community-aware patterns with higher support values. In general, DIVa achieves average improvement over global CASes up to 16% in Google+ datasets.

Furthermore, compared to DIVa extracted CASes, some global CASes are not represented in 20% of the communities detected across all Google+ datasets. Thus, if these communities apply identity validation using global CASes, new users will be penalized for mismatching some validation rules that are unexpressed for the communities they want to join. Figure 5 depicts the penalty ratio paid by some communities when using global CASes against when using DIVa CASes. Namely, these communities represent minority trends discriminated by the universal learning.

D. Complexity Analysis

The model's cost is an aggregation of its steps. First, every node computes its LCAS. The complexity of this is a function of the number of node's friends (i.e., its degree d) and of the number of profile attributes in the profile schema. Indeed, the LCAS learning requires computing for every pair of attributes (a profile schema of m attributes results in $p = \binom{m}{2} = \frac{m^2-m}{2}$ number of pairs), its value-co-occurrence among all the node's direct friends. Therefore, the number of performed checks per attributes pair is, $c = \binom{d}{2} = \frac{d^2-d}{2}$. Accordingly, the LCAS learning's complexity is $\mathcal{O}(c*p)$. By this, the nodes with higher degree would be the bottlenecks in the LCAS learning step; however, this step is node dependent and does not require the simultaneous online availability of all the nodes. Moreover, it is executed only upon significant changes to the network.

regarding the community detection step, it costs in terms of communication traffic between all the nodes in the OSN. By our adopted work for decentralized community detection, this step's cost is $\mathcal{O}(N*D*R)$, where N is the total number of nodes in the OSN graph, D is the average node degree, and R is the total number of rounds needed for the algorithm to converge[5] [19]. This can be very costly for large graphs, but this step is a one time process only.

V. Conclusion and Future Work

In this paper, we have introduced DIVa, a novel unsupervised and incremental learning scheme as a fully decentralized identity validation model for OSNs in contrast to existing global approaches. DIVa conceptualizes user online identities by mining the correlations among user profile attributes not from user population as a whole, but from individual communities, where the correlations are more pronounced. In our experiments we show that reliance on revealing the highly expressed patterns inside communities resulted in extracting

[5] R depends on the topological properties of the underlying graph

community-aware validation rules with average improvements upto 50% than the universal rules that only reveal the global patterns. Furthermore, our model maintains users' privacy during the learning phase as users profiles information are processed only by their direct friends. The experiments show the effectiveness and scalability of our proposed model.

As a natural continuation of the work, we plan to implement a gossip-based protocol to exchange LCAS among nodes in communities instead of depending on diva nodes. We plan to investigate trading off between convergence time with efficiency in terms of scalability and resilience to nodes' failures.

ACKNOWLEDGMENT

This work is under the umbrella of the iSocial EU Marie Curie ITN project (FP7-PEOPLE-2012-ITN). The authors also thank Naeimeh Laleh for her help with Facebook data sanitizing.

REFERENCES

[1] W. Luo, J. Liu, J. Liu, and C. Fan, "An analysis of security in social networks," in *DASC'09*. IEEE, 2009.

[2] M. Huber, M. Mulazzani, E. Weippl, G. Kitzler, and S. Goluch, "Friend-in-the-middle attacks: Exploiting social networking sites for spam," *Internet Computing*, 2011.

[3] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Communications of the ACM*, 2007.

[4] Facebook. (2014) Facebook annual report. [Online]. Available: http://investor.fb.com/annuals.cfm

[5] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, "Decent: A decentralized architecture for enforcing privacy in online social networks," in *PeRCom'12*. IEEE, 2012.

[6] A. C. Squicciarini, C. Griffin, and S. Sundareswaran, "Towards a game theoretical model for identity validation in social network sites," in *PASSAT'11*. IEEE, 2011, pp. 1081–1088.

[7] H. Krasnova, O. Günther, S. Spiekermann, and K. Koroleva, "Privacy concerns and identity in online social networks," *Identity in the Information Society*, 2009.

[8] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *Security and Privacy*. IEEE, 2008.

[9] L. Jin, H. Takabi, and J. B. Joshi, "Towards active detection of identity clone attacks on online social networks," in *CODASPY'11*. ACM, 2011.

[10] B.-Z. He, C.-M. Chen, Y.-P. Su, and H.-M. Sun, "A defence scheme against identity theft attack based on multiple social networks," *Expert Systems with Applications*, 2014.

[11] M. Sirivianos, K. Kim, J. W. Gan, and X. Yang, "Assessing the veracity of identity assertions via osns," in *COMSNETS'12*. IEEE, 2012.

[12] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. S. Kim, P. Compton, and A. Mahidadia, "Collaborative filtering for people to people recommendation in social networks," in *Advances in Artificial Intelligence*. Springer, 2011.

[13] P. Chairunnanda, N. Pham, and U. Hengartner, "Privacy: Gone with the typing! identifying web users by their typing patterns," in *PASSAT'11*. IEEE, 2011.

[14] G. Roffo, C. Segalin, A. Vinciarelli, V. Murino, and M. Cristani, "Reading between the turns: Statistical modeling for identity recognition and verification in chats," in *AVSS'13*. IEEE, 2013, pp. 99–104.

[15] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *WWW'13*. International World Wide Web Conferences Steering Committee, 2013.

[16] L. Bahri, B. Carminati, and E. Ferrari, "Community-based identity validation on online social networks," in *ICDCS'14*. IEEE, 2014, pp. 21–30.

[17] P. N. Krivitsky, M. S. Handcock, A. E. Raftery, and P. D. Hoff, "Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models," *Social networks*, 2009.

[18] E. Ferrara, "Community structure discovery in facebook," *International Journal of Social Network Mining*, 2012.

[19] F. Rahimian, S. Girdzijauskas, and S. Haridi, "Parallel community detection for cross-document coreference," in *IEEE/WIC/ACM*. IEEE, 2014.

[20] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, 2012.

[21] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD*. ACM, 1993.

[22] A. Soliman, L. Bahri, B. Carminati, E. Ferrari, and S. Girdzijauskas, "Diva: Decentralized identity validation for social networks," KTH-Royal Instituite of Technology, Tech. Rep., 2015. [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:805740/FULLTEXT02.pdf

[23] H. Meyerhenke, B. Monien, and T. Sauerwald, "A new diffusion-based multilevel algorithm for computing graph partitions of very high quality," in *IPDPS'08*. IEEE, 2008.

[24] P. Sanders and C. Schulz, "Distributed evolutionary graph partitioning." in *ALENEX*. SIAM, 2012.

[25] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *SIGCOMM'6*, vol. 36, no. 4. ACM, 2006, pp. 267–278.

[26] Y. Boshmaf, K. Beznosov, and M. Ripeanu, "Graph-based sybil detection in social and information systems," in *ASONAM'13*. IEEE, 2013.

[27] C. G. Akcora, B. Carminati, and E. Ferrari, "Privacy in social networks: How risky is your social graph?" in *ICDE'12*. IEEE, 2012, pp. 9–19.

[28] N. Z. Gong, W. Xu, L. Huang, P. Mittal, E. Stefanov, V. Sekar, and D. Song, "Evolution of social-attribute networks: measurements, modeling, and implications using google+," in *Internet Measurement Conference*. ACM, 2012, pp. 131–144.