GALG: Linking Addresses in Tracking Ecosystem Using Graph Autoencoder with Link Generation

Tianyu Cui[1,2], Gang Xiong[1,2], Chang Liu(✉)[1,2,⋆], Junzheng Shi[1,2], Peipei Fu[1,2], and Gaopeng Gou[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
{cuitianyu, xionggang, liuchang, shijunzheng, fupeipei, gougaopeng}@iie.ac.cn

Abstract. Online tracking technology is a critical tool for user-centric platform practitioners to link users across multiple web pages and make detailed user profiles for the improvement of recommender systems like targeted advertising. Recently, due to the dynamic address allocation and security upgrade, mitigations indirectly make prior tracking techniques unreliable. To overcome the problem, traffic-based tracking techniques are proposed to link users' dynamic addresses through similarity learning of user behaviors in their traffic interaction. However, prior work either provides poor similarity learning ability or is impractical when applied to a large scale. In this paper, we propose GALG, a graph-based artificial intelligence approach to link addresses for user tracking on TLS encrypted traffic. GALG uses the framework of graph autoencoder and adversarial training to learn the user embedding with semantics and distributions. Employing a new theory – link generation, GALG could link all the addresses of target users based on the knowledge of address-service links. When evaluated on real-world user datasets, GALG outperforms existing approaches in both performance and practicality.

Keywords: Online tracking · Graph neural networks · Link prediction

1 Introduction

Websites and third-party services such as search engines, advertising networks, and network providers collect user interests across multiple web pages to improve the quality of recommender systems and user experiences in the area including targeted advertising and content personalization. Under the background, online tracking has been ubiquitous on the web [3]. The tracking mechanism could link records of a user's browsing activity across numerous websites to make inferences about the user's demographics and interests, or observe the conversion that whether an advertisement on a website leads to the desired user activity on another website [17]. Until recently, over four-fifth of websites have enabled tracking systems [25]. Big players like Google and Facebook leverage the widespread

⋆ Chang Liu is the corresponding author.

use of their advertising networks and social plugins to track users across websites and gain detailed user profiles.

The core objective of online tracking is to link historical users across multiple contexts. Refer to Figure 1. When a user generates browsing records on a website, the tracking system could utilize the information in the incoming records to search all records linking to the user in a tracking database. The database is a knowledge base owned by a tracker that contains numerous user records on multiple websites. The linked records of each user are finally used in user profiling for a better recommendation. In this setting, the traceable information in records is critical for the tracking system.

Promoted by the huge tracking ecosystem, researchers have tried to leverage types of information to track real-world users, including IP addresses, cookies, and browser fingerprints. Tracking based on IP addresses is the primary tracking method because all online behaviors must come from users' client addresses. However, the dynamic address allocation causes frequent changes of users' addresses, making address-to-user correlation unreliable. Tracking based on cookies [19] and browser fingerprints [7] could produce and store the identifier and features of user browsers. Nevertheless, as users are increasingly aware of privacy protection, they start to encrypt communication sessions with Transport Layer Security (TLS) [20], use private browsing modes, and enable privacy-friendly browser extensions to obfuscate traceable features. Although the majority of users only intend to protect their sensitive information rather than deliberately confronting recommender systems, the situation indeed leads to data loss and model failure in the tracking ecosystem. Microsoft reported that they could no longer track 32% of users under their services due to these mitigations [24].

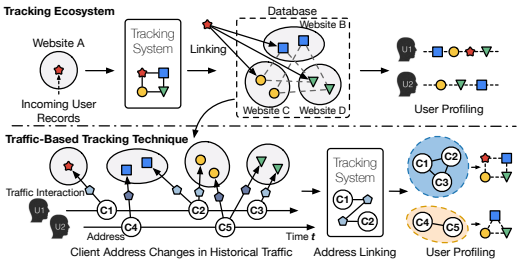


Fig. 1. The illustration of the tracking ecosystem. The traffic-based tracking method links users' addresses across multiple traffic interactions.

To address these problems, recently, traffic-based user tracking techniques [1, 4, 8, 14] have been proved to have a strong performance by analyzing the patterns in the traffic. As a worldwide information system, the Internet maintains users' daily online activities through traffic transmission. Though the critical payload consisting of user data is encrypted in the TLS session under HTTPS communication, the traffic contains considerable meta-information associated with user behaviors and online interaction. In Figure 1, banding the address with these traffic characteristics, researchers could link multiple client addresses to achieve long-term user tracking. However, the extensive knowledge hidden in the traffic raises the questions that how to effectively leverage the complicated

information to reach high tracking accuracy, and, separately, how to fast link the real-world users in such a huge knowledge volume. Previous works either could only track a specific subset of users on a closed-world dataset due to the unreliable similarity learning [1, 8, 14], or expense considerable time with an unsuitable framework [4], remaining the problem that the traffic-based tracking technique is impractical when applied to a large scale.

In this paper, we develop a more sophisticated approach to overcome these limitations, juggling performance and practicality. In particular, we introduce **GALG**, **a graph-based artificial intelligence approach** to link addresses for user tracking on TLS encrypted traffic. GALG is short for "**G**raph **A**utoencoder with **L**ink **G**eneration". The framework of our approach consists of three steps as follows. First, GALG constructs a graph with client address-to-online service links and the user preference distribution of each client address to model the traceable information. Second, for better similarity learning, using the theory of Graph Neural Networks [12, 13, 18, 23] and adversarial training, GALG employs a stack attention-based encoder and a discriminator to learn the latent embedding of jointing semantics and distributions. Third, innovating from the task of link prediction [16] in social networks, we propose **link generation**, which could learn from address-service links to generate address-address links. Benefiting from the new theory, GALG could achieve effective few-shot learning, finally taking large-scale user tracking to reality.

In summary, our contributions can be summarized as follows:

- **Application.** We implement reliable user tracking in the tracking ecosystem to link addresses through address embedding learning with a traffic-based tracking system.
- **Theory.** We propose link generation to generate a new type of links from the original type of links in heterogeneous graphs, which is more effective in user tracking tasks than link prediction.
- **Models.** We introduce two novel tracking models GALG and VGALG, which could jointly learn semantics and distributions through an adversarial architecture with a graph autoencoder.
- **Experiments.** We conduct experiments on real-world datasets. Results indicate that GALG outperforms the state-of-the-art tracking techniques and link prediction-based methods in both performance and practicality.

2 Related Work

We overview the related work of our paper from the objective and technical perspectives, including the prior work of user tracking and link prediction.

2.1 User Tracking

User tracking on the Internet could come in various forms. Over the past years, through HTTP cookies [19] and browser fingerprinting [7]. However, recently,

traffic-based tracking techniques have been proved to have stronger performance and longer tracking time, which can be extensively applied in various scenarios. Kumpost et al. [14] used the target IP addresses of each user to build user profiles for tracking them in the future. Banse et al. [10] trained a Bayesian classifier with DNS requests to track users on a university network. Following the deployment of HTTPS, the interaction between users and websites tends to be protected under the wide-used TLS traffic. Anderson et al. [1] extracted field values in TLS ClientHello messages to build traceable fingerprints, which could be easily enhanced with machine learning. Nonetheless, due to the strong variation of different users in the open-world scenarios, these approaches usually perform a poor generalization on unseen users. To address the problem, Cui et al. [4] proposed a knowledge graph-based approach SiamHAN to track users on TLS encrypted traffic. Since the approach requires calculating the similarity between every two addresses, the time cost is impractical at scale. Different from prior work, GALG aims to provide a better learning framework of user embeddings, which could work on unseen users with less time cost.

2.2 Link Prediction

Link prediction [16] is a critical task for graph-structured data. By predicting the relationship of two nodes in a graph, the task has many applications such as recommendation and graph reconstruction. The prior approaches for link prediction mainly include heuristic methods, latent feature methods, and explicit feature methods. Heuristic methods [16] are a class of simple yet effective approaches to calculating node similarity scores with heuristic assumption. Latent feature methods such as spectral clustering [21] and node2vec [9], have been proposed to use the knowledge of graph structure for graph embedding learning. Using the powerful performance of GNNs [12, 13, 18, 23] , explicit feature methods [12, 18] could aggregate node attributes built from side information to obtain more meaningful knowledge from the graph. While the techniques for link prediction have many variants, the theory of the task is never changed – using the known links to predict the same type of unknown links. The framework limits that link prediction requires learning considerable annotation links to supplement the lost links in one specific graph. In the tracking database, there is a large ratio of test user nodes and we cannot always keep enough labels in a graph. In this paper, we employ link generation to overcome the limitation in the tracking ecosystem.

3 Preliminaries

This section introduces the definition of the problem and the link generation theory to help readers understand this paper.

3.1 Problem Definition

On the Internet, users could use clients to access various online services. The manifestation in the traffic is that the client address and the server address es-

tablish connections. However, to facilitate the address acquisition process, network administrators have widely deployed dynamic address allocation policies like DHCP [5]. Within a period of time, a user might use multiple client addresses for external communication. The relationship between users and addresses cannot be detected using payloads due to TLS encryption.

In the tracking ecosystem, a tracker aims to link user across multiple web pages to make detailed user profiles. To obtain the user traffic, the tracker could be a network provider with a vantage traffic observation point, an advertising network provider with wide-used traffic plugins deployed in numerous Apps and websites, or a content provider like Google owning multiple websites. Using the meta-information in the traffic, tracking systems could link multiple client addresses to find out the target user. Given a period of TLS historical traffic as background knowledge K_t, the set of all client addresses in the traffic is S, the set of client addresses of a target user is $Y = \{y_0, y_1, ..., y_n\}$. A tracking system F could use one client address y_0 to trace the whole address set Y:

$$F((S, y_0)|K_t; \theta) = Y \tag{1}$$

where θ is the parameters of the tracking model. After obtaining the address set Y, researchers could master all activities associated with these addresses to analyze the target user. Holding the long sequences across websites, trackers could infer global user demographics for service policy updates or use the interaction records between addresses and accounts for targeted recommendations.

3.2 Link Generation

The link generation task we proposed is innovated from the link prediction.

Definition 1. Link Prediction. Given a graph G, which contains at least one type of node and one type of link. The link prediction task requires learning a type of links and complementing the missing links of this type in graph G. If A_i is the adjacency matrix of the links with type i, the goal of a model F for link prediction could be shown as follows:

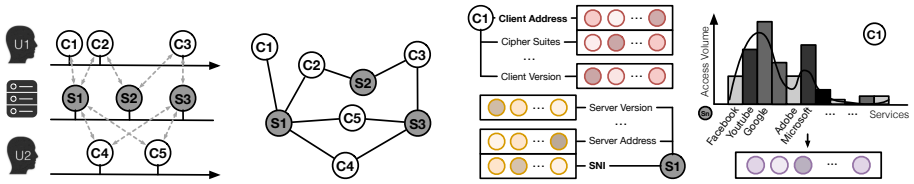
$$A_i \xrightarrow{F(G;\theta)} \widetilde{A_i} \tag{2}$$

where θ is the trained parameters. $\widetilde{A_i}$ is the ground truth of the type-i links.

Definition 2. Link Generation. Given a heterogeneous graph G_h, which contains at least two types of nodes and one type of link. The link generation task requires learning a type of links and generating a new type of links in graph G_h. If A_i is the adjacency matrix of the links with type i, the goal of a model F for link generation could be shown as follows:

$$A_i \xrightarrow{F(G_h;\theta)} A_j \tag{3}$$

where $i \neq j$. A_j is the ground truth of the type-j links. Unlike link prediction, link generation tasks can generate types of links that never appear in the graph. In this paper, we show that link generation is more effective and practical than link prediction by using address-service links to generate address-address links.



(a) user behaviors with client addresses. (b) address-service graph construction. (c) client and service fingerprints. (d) user preference distributions.

Fig. 2. The address-service graph and user preference distributions: (a) Users might use multiple client addresses (c1, c2, and c3 for user u1; c4 and c5 for user u2) to access online services (s1, s2, and s3) in a period of time; (b) The connection relationship between c and s could build the address-service graph; (c) Each node c/s uses client/service fingerprints built from TLS traffic as the node attribute; (d) The cumulative visits of a client address to each service could build the preference distribution.

4 Design of GALG

This section proposes the overall framework of GALG, including (1) graph and distribution construction and (2) GALG's model architecture.

4.1 Graph and Distribution Construction

To implement traffic-based tracking technology, GALG extracts two kinds of knowledge from the traffic to help track real-world users – an address-service graph and user preference distributions.

Address-service Graph To model the user activities behind the traffic, GALG uses a heterogeneous graph to capture the meta-information in the traffic. Figure 2(a)-(c) shows the detail of building the address-service graph. In the historical traffic over a period of time, since users use multiple client addresses to access online services, the connection relationship between these addresses and services could be used to build the heterogeneous graph. The graph contains two types of nodes and one type of link – address node c, service node s, and address-service link c-s. Whenever a user accesses a web service over HTTPS, their communication will generate many available data in TLS traffic such as ClientHello, ServerHello, and Certificate message. GALG extracts client fingerprints and service fingerprints from these messages to model the attributes of the address nodes and the service nodes in the graph. Table 1 shows the notions of these fingerprints. In each TLS connection, the fingerprints are bound to an address node and a service node respectively. The client address and the server name identifier (SNI) are used as the node identifier to distinguish different address nodes and service nodes. Finally, these node attributes are learned by doc2vec [15] to obtain the semantic representation of fingerprints under the feature space.

Table 1. The client fingerprints and the service fingerprints used to build the attributes of address nodes and service nodes. Different address nodes or service nodes are distinguished by the client address or the server name identifier (SNI).

Type	Fingerprint Name	Notion	Label
Address node c	Client address*	The address of the client in the TLS traffic	Fc1
	Record version	The TLS version employed by the client	Fc2
	Client version	The version by which the client wishes to connect	Fc3
	Cipher suites	A list of the cryptographic options supported	Fc4
	Compression	A list of the compression methods supported	Fc5
Service node s	SNI*	The domain name that the client wants to reach	Fs1
	Server address	The address of the server in the TLS traffic	Fs2
	Record version	The TLS version employed by the server	Fs3
	Server version	The version finally chosen by the server	Fs4
	Cipher suite	The single cipher suite selected	Fs5
	Algorithm ID	The identifier for the cryptographic algorithm	Fs6
	Issuer	The entity that has signed and issued the certificate	Fs7
	Subject	The entity associated with the public key stored	Fs8

User Preference Distributions In addition to knowing which services the user accessed in the horizontal analysis, a vertical eye to master how much the user prefers these services could also contribute to identifying the user. GALG uses the cumulative access volume of a client address to each service to build its user preference distribution. Figure 2(d) shows the detail of building the distribution. For each client address, we collect the number of TLS connections of the address to each service to build a distribution vector. The length of the vector is the total number of service nodes in the address-service graph. Finally, to reduce the overlong dimension of the vector, GALG employs PCA [6] to obtain the representation of the distribution for each address node in the graph.

4.2 Model Architecture

Figure 3 shows the overall architecture of GALG. GALG employs an adversarial architecture with a graph autoencoder to implement reliable user tracking.

Encoder Learning In the task of user tracking, the heterogeneous graph G_h built from TLS traffic contains address-service links A and node attributes X. By learning the meta-information, the goal of the encoder in GALG is to obtain the latent embedding Z of the address nodes for address-address link generation.

To implement the encoder model, GALG leverages stack attention to integrate complex semantic knowledge into the embeddings of the address nodes. The stack attention contains two levels – fingerprint-level attention (FA) and service-level attention (SA). The fingerprint-level attention first learns the weights of all fingerprints in a node attribute X_u and aggregates them to obtain the first-level node embedding. The fingerprint importance α_u and the first-level node

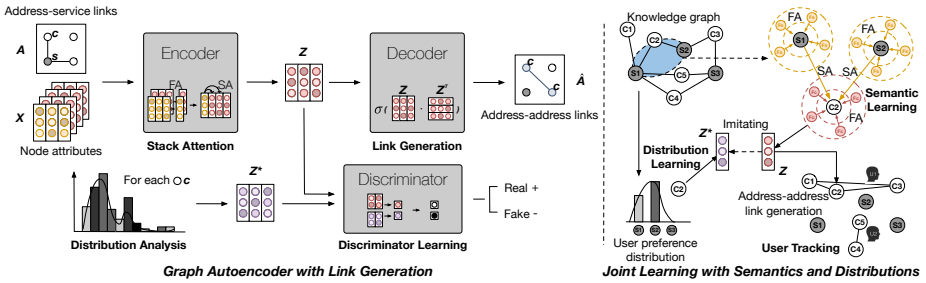


Fig. 3. The overall architecture of GALG. The encoder aggregates meta-information in the knowledge graph to obtain the latent embedding through stack attention. The discriminator distinguishes between the representation of real preference distributions and the latent embedding. The decoder finally generates the address-address links.

embedding Z_u^1 of a node u are shown as follows:

$$\alpha_{ui} = \frac{\exp(h_{ui}^\top W_h)}{\sum_{i=1}^{|X_u|} \exp(h_{ui}^\top W_h)}, \text{ here } h_{ui} = \tanh(W_w X_{ui} + W_b)$$

$$Z_u^1 = \sum_{i=1}^{|X_u|} \alpha_{ui} X_{ui} \tag{4}$$

where W_w, W_b, and W_h are the parameter matrices. X_{ui} is a client fingerprint of an address node c or a service fingerprint of a service node s in the heterogeneous graph G_h. Using these fingerprints, the fingerprint attention aims to learn the unique client or service representation from the client or service profiles.

The service-level attention then employs a Graph Attention Network (GAT) [23] based approach to learn the latent embedding of the address nodes. For an address node c_i, S_{c_i} includes node c_i and the service nodes linked to it. The service-level attention could calculate the importance of the services to identifying the user behind the address node c_i and aggregates them to obtain the latent embedding $Z_{c_i}^2$ of the address node:

$$\beta_{c_i s_j} = \frac{\exp(h_{c_i s_j})}{\sum_{s_j \in S_{c_i}} \exp(h_{c_i s_j})}, \text{ here } h_{c_i s_j} = \text{LeakyReLU}(W_s[Z_{c_i}^1 \| Z_{s_j}^1])$$

$$Z_{c_i}^2 = \Big\|_{k=1}^{K} \text{ELU}\Big(\sum\nolimits_{s_j \in S_{c_i}} \beta_{c_i s_j} Z_{s_j}^1\Big) \tag{5}$$

where W_s is the parameter matrix. $\|$ represents the concatenation operation. K is the number of heads using the multi-head attention mechanism [22]. Using the first-level embeddings, the service-level attention aims to capture the semantics of the user behavior through the communication relationships.

Finally, the stack attention of GALG realizes stacking the two-level semantics into the latent embedding. Through semantic learning, the three-layer meta-information of fingerprints, services, and client addresses are orderly squeezed in

the final representation. We collect the latent embedding of all address nodes in the graph to form the latent embedding Z:

$$Z = \prod_{c_i \in V_c} Z_{c_i}^2 \tag{6}$$

where V_c is the set of address nodes in the graph G_h. Similar to previous work [12, 18], GALG could be extended to a variational autoencoder version VGALG for link generation. The encoder of VGALG is defined as follows:

$$Z = \prod_{c_i \in V_c} \mathcal{N}(z_{c_i} | \mu_{c_i}, \mathrm{diag}(\sigma^2)) \tag{7}$$

where $\mu = Z^2$ and $\log \sigma = Z^{2'}$ are the matrices of the mean and log variance vectors output by two stack attention networks. Using the two vectors, the model is modified to sample the latent embedding Z to improve the robustness.

Decoder Learning Using the latent embedding Z, GALG's decoder aims to generate the address-address links $\hat{A}$ for tracking. We could predict whether two client addresses belong to the same user by judging whether there is a link between the two address nodes. The work is implemented by an inner product between their latent embeddings:

$$\hat{A}_{ij} = \prod_{c_i \in V_c} \prod_{c_j \in V_c} \mathrm{sigmoid}(Z_{c_i}^\top Z_{c_j}) \tag{8}$$

where c_i and c_j are two address nodes in the graph G_h.

Adversarial Training To model the complex user behaviors behind the client addresses, in addition to the semantic knowledge, GALG is also required to use the distribution knowledge to fully grasp the users' activities. Using a multi-layer perceptron (MLP) based discriminator, GALG employs adversarial training to embed the distribution knowledge into the latent embedding Z.

Through the distribution analysis for each address node c, GALG obtain the user preference distribution Z^*. The goal of discriminator is to distinguish whether an input is from the prior distribution or from GALG's encoder.

During the adversarial training, GALG's discriminator aims to identify the real distribution and classify the latent embedding into the fake class. Therefore, we could optimize the discriminator by minimizing the cross-entropy cost J_D:

$$J_D = -\mathbb{E}_{z \sim p_z} \log D(Z^*) - \mathbb{E}_{z \sim F_{\mathrm{en}}} \log(1 - D(Z)) \tag{9}$$

where p_z is the real user preference distribution formed by all client addresses in the historical TLS traffic. $D(Z)$ is the discrimination score. To deceive the discriminator, in addition to minimizing the error between the ground truth A^* and the adjacency matrix $\hat{A}$ generated from the latent embedding Z, the encoder is also required to imitate the real preference distributions. Therefore, the cost of the encoder J_E during the adversarial training could be defined as follows:

$$J_E = \mathbb{E}_{\hat{A} \sim F_{\mathrm{en}}} \log |\hat{A} - A^*| - \mathbb{E}_{z \sim F_{\mathrm{en}}} \log D(Z) \tag{10}$$

Table 2. The composition of graphs built from three datasets.

Dataset	Nodes c	Nodes s	Links c-s	Users
P-AllService	1,016	5,597	7,022	450
P-Google	723	313	4,134	550
CSTNET	958	5,517	6,840	685

where the ground-truth adjacency matrix A^* only contains links between training nodes. The encoder cost J'_E of the variational autoencoder variant VGALG could also be defined as follows:

$$J'_E = \mathbb{E}_{\hat{A}\sim F_{\text{en}}} \log|\hat{A} - A^*| - \mathbb{E}_{z\sim F_{\text{en}}} \log D(Z) - \text{KL}[Z\|q(z)] \tag{11}$$

where KL is the Kullback-Leibler divergence. $q(z) = \prod_{c_i} \mathcal{N}(z_{c_i}|0, \text{I})$ is a Gaussian prior that we followed Kingma et al. [11]. Finally, we use the adjacency matrix $\hat{A}$ to track users in the tracking ecosystem.

5 Experiment Setup

Datasets In this work, our evaluation datasets consist of a public dataset CSTNET and two participant datasets P-AllService and P-Google generated from 1k participants in two months in our experiments. Table 2 provides the graph composition built from the three datasets.

(1) **CSTNET.** CSTNET is a public dataset collected from March to July 2018 on China Science and Technology Network. Cui et al. [4] monitored the traffic on a vantage point to achieve tracking on IPv6 networks. In the network, 80% of IPv6 users change their client addresses at least once a month. We use the dataset to track real-world users from the perspective of a network provider.

(2) **P-AllService and P-Google.** To conduct extensive experiments, we invited 1k participants to join the traffic collection work under mobile networks. We installed the traffic plugin in participants' devices to record their daily online behaviors with consent. The participants are divided into two groups. For one group, we recorded all the TLS interaction traffic generated by participants to imitate a third party who tracked users with traffic plugins deployed on numerous Apps and build the P-AllService dataset. For the other group, the participants are required to access Google services following their online habits. These activities cannot be fully tracked by Google accounts since many services or web pages are not required to log in for browsing, such as Google Scholar and blogs. We recorded the traffic to form the P-Google dataset to track users from the perspective of a content provider like Google.

Baselines The baselines in our experiments for comparison mainly include representative link prediction approaches and tracking techniques.

(1) **Link Prediction Approaches.** We compare types of link prediction approaches in this paper. **Common Neighbors (CN)** [16], **Jaccard** [16], and

Preferential Attachment (PA) [2] are three heuristic methods to determine a link between two nodes. **Spectral Clustering (SC)** [21] and **node2vec** [9] are two latent feature methods to learn graph embeddings. **GAE** [12], **VGAE** [12], **ARGA** [18], and **ARVGA** [18] are four explicit feature methods to aggregate node features through graph autoencoder or its variational version.

(2) **Tracking Techniques.** We implement four representative tracking techniques, which use multiple characteristics in the TLS traffic. **User IP Profiling** [14] and **User SNI Profiling** [8] are methods to build user profiles through the destination IPs of the client addresses or the SNIs and track users through a Bayesian classifier. **Client Fingerprinting** [1] is a method to extract fields in ClientHello messages as client fingerprints and learn the fingerprints through Random Forest. **SiamHAN** [4] is a method to build graphs for each client address and learn the similarity of each two graphs through siamese networks.

Implementation During the data preprocessing, we limit the maximum fingerprint length to 50. To train the doc2vec model, we set the vector size as 50 and the window size as 5 to obtain the representations of the fingerprints. The output dimension of PCA is 32 for the representations of the distributions. When training GALG, we randomly initialize parameters and optimize the model with the Adam algorithm. The learning rate is set as 0.005. The number of e-steps is 5 and the number of d-steps is 1. The number of attention head K is 4. We use four metrics including TPR, FPR, AUC, and AP to evaluate the models.

6 Evaluation

This section presents all experimental results to implement online user tracking.

6.1 Distribution Analysis

We provide the result of distribution analysis to indicate the effectiveness to leverage the distribution knowledge. Figure 4 shows the representation of distributions in P-AllService dataset. Results indicate that the preference distributions of the client addresses belonging to the same user are similar, demonstrating that exploiting this knowledge could help distinguish the client addresses of different users to a certain degree. For instance, `Address0` and `Address1` keep similar distribution representations because they both belong to `User0`. While the representation of `Address2` belonging to `User1` is obviously different from the former

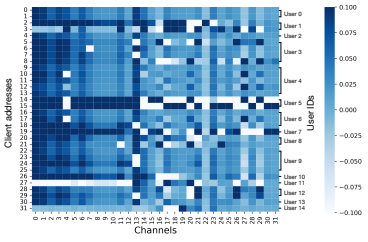
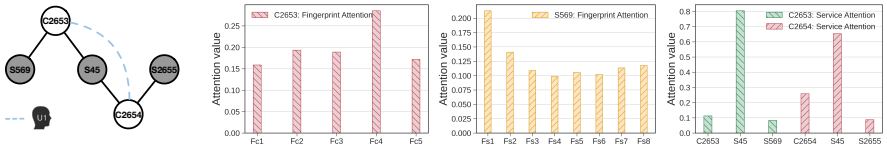


Fig. 4. The preference distributions of client addresses coming from 15 users.



(a) a graph case for attention analysis. (b) c2653's fingerprint attention. (c) s569's fingerprint attention. (d) service attention of c2653 and c2654.

Fig. 5. A case study of stack attention to help track users through semantic knowledge.

addresses in visual. With deeper analysis, we find that `Address0` and `Address1` have accessed many common domains with similar visit volumes, including `google`, `cloudflare`, `eroimg`, and `share-videos`. While `Address2` has never accessed these domains. GALG could embed the distribution knowledge into the embedding to help link the client addresses.

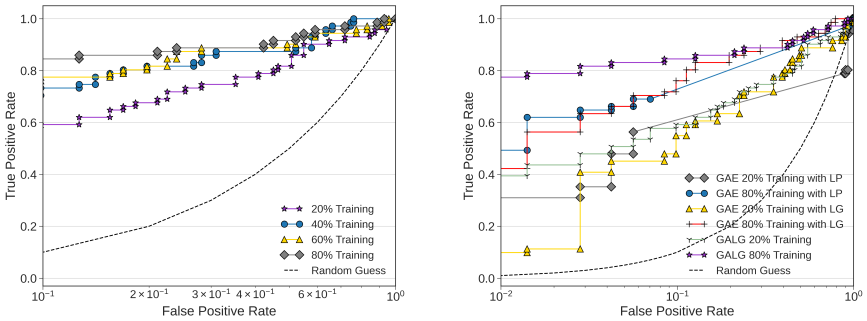
6.2 Attention Analysis

Figure 5(a) shows a knowledge graph case, including two address nodes (`c2653` and `c2654`), three service nodes (`s569`, `s45`, and `s2655`), and the address-service links between them. In this setting, `c2653` and `c2654` both belong to the user `u1`. Figure 5(b) and Figure 5(c) show the fingerprint attention of the address node `c2653` and the service node `s569` respectively. The label corresponding to each fingerprint is shown in Table 1. For the client fingerprints of `c2653`, `Fc4` contributes more to the task obviously since it is the significant browser parameter that could be used to identify the client used by a user. Due to the change of the client address, `Fc1` surely obtains the lowest attention for tracking users. For the service fingerprints of `s569`, `Fs1` becomes the critical service fingerprint to indicate the attribute of the service accessed by users. The fingerprint-level attention finally obtains unique client and service embeddings through learning semantic information. The service attention of `c2653` and `c2654` is illustrated in Figure 5(d). The high attention value of `s45` indicates that service attention could find the same service accessed by both two client addresses to help link them to the same user. Finally, the service-level attention could learn the semantics of address-service communication and obtain the meaningful embeddings.

6.3 Link Generation

To explore the effectiveness, we first measure the link generation performance by evaluating the correctness of the links generated between the test nodes.

Few-shot Learning In Figure 6(a), we show GALG's performance on the P-AllService dataset with different ratios of training users. With only 20% training users, GALG could obtain an acceptable performance with 81% AUC. This advantage comes from the graph structure of address-service links to propagate the



(a) training ratios in GALG. (b) advantages of link generation.

Fig. 6. The link generation performance of GALG on different training ratios and the advantage of link generation (LG) compared to link prediction (LP) methods.

knowledge of labeled address nodes to the other non-labeled address nodes. Finally, GALG could achieve 92% AUC with an 80% training ratio. To demonstrate the advantage of link generation, we implement two frameworks of a representative model GAE under link prediction and link generation. Figure 6(b) shows the performance of the two frameworks on 20% and 80% training ratios compared with GALG. With 20% training users, the link prediction method could only obtain 55% AUC. However, with the link generation framework, GAE could achieve 75% AUC under this training ratio. Since link prediction methods require learning the graph with address-address links, when limiting the number of the labeled address nodes, the graph will lose considerable links to propagate the label knowledge. For link generation methods, since the address-service links of each address node are easy to obtain regardless of whether the nodes are labeled, the always complete graph structure help link generation more effective.

Overall Performance Finally, we modify link prediction baselines with the link generation framework and show the performance of all link prediction baselines under the frameworks of link prediction and link generation in Table 3. Results indicate that the performance of link generation is better than link prediction for all baselines on the three user datasets. It demonstrates that learning the online behaviors behind the address-service links is more effective than inferring the neighbor relationships through address-address links. Our two models GALG and VGALG outperform all baselines using the theory of link generation.

6.4 User Tracking

To test the tracking performance, we set one address node of each test user as the known nodes to evaluate whether we could link all address nodes belonging to the same users with the known nodes.

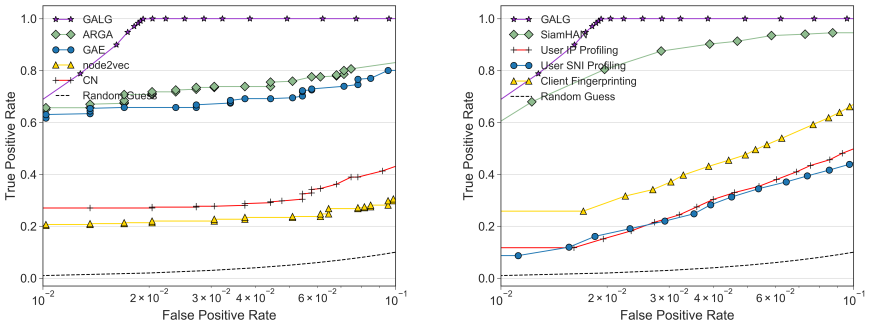
Table 3. The overall performance of all link prediction baselines under the frameworks of link prediction and link generation.

Method	Link Prediction						Link Generation					
	CSTNET		P-AllService		P-Google		CSTNET		P-AllService		P-Google	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
CN	0.528	0.515	0.619	0.606	0.600	0.596	0.632	0.617	0.623	0.615	0.728	0.724
Jaccard	0.502	0.495	0.610	0.605	0.582	0.577	0.619	0.611	0.632	0.606	0.759	0.763
PA	0.524	0.524	0.470	0.508	0.595	0.653	0.526	0.557	0.593	0.594	0.603	0.661
SC	0.648	0.680	0.490	0.641	0.396	0.547	0.661	0.687	0.615	0.686	0.589	0.635
node2vec	0.511	0.516	0.514	0.532	0.444	0.476	0.516	0.539	0.633	0.686	0.626	0.641
GAE	0.888	0.887	0.840	0.850	0.933	0.942	0.978	0.985	0.920	0.942	0.941	0.955
VGAE	0.842	0.840	0.850	0.857	0.893	0.901	0.970	0.977	0.929	0.946	0.930	0.951
ARGA	0.918	0.924	0.915	0.937	0.889	0.931	0.938	0.949	0.934	0.945	0.940	0.950
ARVGA	0.867	0.879	0.881	0.909	0.881	0.887	0.886	0.889	0.916	0.936	0.902	0.904
GALG	-	-	-	-	-	-	0.980	0.984	**0.957**	**0.969**	0.940	0.959
VGALG	-	-	-	-	-	-	**0.988**	**0.990**	0.937	0.954	**0.965**	**0.974**

Table 4. The overall tracking performance and the inference time of all tracking techniques to predict 1 million link relationships from 0.5k users

Method	CSTNET		P-AllService		P-Google		One	Total
	AUC	AP	AUC	AP	AUC	AP	Inference	Time
User IP Profiling	0.563	0.001	0.611	0.000	0.596	0.000	0.0006s	623.0840s
User SNI Profiling	0.611	0.001	0.645	0.000	0.535	0.000	0.0005s	592.0129s
Client Fingerprinting	0.790	0.005	0.740	0.001	0.755	0.001	0.0091s	9157.4099s
SiamHAN	0.948	0.723	0.967	0.824	0.976	0.727	0.0013s	1323.6057s
GALG - Attention	0.967	0.230	0.968	0.479	0.949	0.201	**0.1604s**	**0.1604s**
GALG - Distribution	0.990	0.890	0.975	0.889	0.966	0.826	0.1893s	0.1893s
GALG	0.995	**0.925**	0.981	0.906	**0.982**	**0.911**	0.1923s	0.1923s
VGALG	**0.996**	0.896	**0.982**	**0.926**	0.971	0.844	0.2030s	0.2030s

Overall Tracking Performance The overall tracking performance of all existing tracking techniques is shown in Table 4. Results indicate that the former three methods obtain bad performance on the AP metric since they can not be applied to the open-world dataset to track the users who are not in the training set. When replacing the stack attention layer in the encoder with a 1-d convolutional layer and a graph convolutional layer (GALG - Attention) or removing the discriminator in GALG (GALG - Distribution), the performance drastically degrades. Compared with the state-of-the-art tracking approach SiamHAN, GALG and VGALG could outperform the method by significant margins. For a deeper analysis, Figure 7(a) and Figure 7(b) show the ROC curves of GALG, link generation-based methods, and existing tracking techniques. Using the link generation framework, explicit feature methods like GAE and ARGA could reach the similar performance of SiamHAN. For a target FPR $= 2\times10^{-2}$, GALG could provide a TPR of 0.99. To explore the inference time, we measure the perfor-



(a) link generation approaches. (b) tracking techniques.

Fig. 7. The tracking performance of the state-of-the-art tracking techniques and the link prediction baselines with the link generation framework.

mance on a single GeForce GTX 1080 Ti GPU. Since GALG is to output the whole adjacency matrix, the inference time for one link is equal to the time for all links. However, prior techniques expense considerable time to infer links between every two nodes in the graph. Therefore, the advantage of our framework helps GALG track 0.5k users in 1 second, which is dramatically faster than the state-of-the-art approach SiamHAN in half an hour.

7 Conclusion

In this work, we explore the implementation to track users on TLS encrypted traffic. We propose GALG, a graph-based artificial intelligence approach to link changed client addresses for finding out the target user. Using the adversarial architecture with a graph autoencoder, GALG could jointly learn the user embedding with semantics and distributions. With a new theory - link generation, GALG could more effectively infer address-address links than the framework of link prediction. Extensive experiments indicate that the performance of our models outperform state-of-the-art methods by significant margins. We published the source code of GALG at `https://github.com/CuiTianyu961030/GALG`.

Acknowledgements This work is supported by the National Key Research and Development Program of China No. 2020YFE0200500 and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02040400.

References

1. Anderson, B., McGrew, D.A.: OS fingerprinting: New techniques and a study of information gain and obfuscation. In: CNS. pp. 1–9 (2017)

2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science (5439), 509–512 (1999)
3. Bashir, M.A., Farooq, U., Shahid, M., Zaffar, M.F., Wilson, C.: Quantity vs. quality: Evaluating user interest profiles using ad preference managers. In: NDSS (2019)
4. Cui, T., Gou, G., Xiong, G., Li, Z., Cui, M., Liu, C.: SiamHAN: IPv6 address correlation attacks on TLS encrypted traffic via siamese heterogeneous graph attention network. In: USENIX Security. pp. 4329–4346 (2021)
5. Droms, R.E.: Dynamic host configuration protocol. RFC 2131 pp. 1–45 (1997)
6. Dunteman, G.H.: Principal components analysis (1989)
7. Gómez-Boix, A., Laperdrix, P., Baudry, B.: Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In: WWW. pp. 309–318 (2018)
8. Gonzalez, R., Soriente, C., Laoutaris, N.: User profiling in the time of HTTPS. In: IMC. pp. 373–379 (2016)
9. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD. pp. 855–864 (2016)
10. Herrmann, D., Banse, C., Federrath, H.: Behavior-based tracking: Exploiting characteristic patterns in DNS traffic. Computer Security pp. 17–33 (2013)
11. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
12. Kipf, T.N., Welling, M.: Variational graph auto-encoders. CoRR (2016)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
14. Kumpost, M., Matyas, V.: User profiling and re-identification: Case of university-wide network analysis. In: TrustBus. pp. 1–10 (2009)
15. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. pp. 1188–1196 (2014)
16. Liben-Nowell, D., Kleinberg, J.M.: The link-prediction problem for social networks. Journal of the American society for information science and technology (7), 1019–1031 (2007)
17. Mayer, J.R., Mitchell, J.C.: Third-party web tracking: Policy and technology. In: SP. pp. 413–427 (2012)
18. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. In: IJCAI. pp. 2609–2615 (2018)
19. Papadopoulos, P., Kourtellis, N., Markatos, E.P.: Cookie synchronization: Everything you always wanted to know but were afraid to ask. In: WWW. pp. 1432–1442 (2019)
20. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. RFC 8446 pp. 1–160 (2018)
21. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Mining and Knowledge Discovery (3), 447–478 (2011)
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS. pp. 5998–6008 (2017)
23. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
24. Yen, T., Xie, Y., Yu, F., Yu, R.P., Abadi, M.: Host fingerprinting and tracking on the web: Privacy and security implications. In: NDSS (2012)
25. Yu, Z., Macbeth, S., Modi, K., Pujol, J.M.: Tracking the trackers. In: WWW. pp. 121–132 (2016)