# Constructing Browser Fingerprint Tracking Chain Based on LSTM Model

Xiaoyun Li
Beijing University of Posts and Telecommunications
Beijing, China
lxy691219491@bupt.edu.cn

Xiang Cui
Cyberspace Institute of Advanced Technology
Guangzhou University
Guangzhou, China
cuixiang@iie.ac.cn

Limin Shi
Shandong University of Science and Technology
Qingdao, China
shilimin410@163.com

Chaoge Liu
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
liuchaoge@iie.ac.cn

Xiaoxi Wang
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
wangxiaoxi@iie.ac.cn

***Abstract*—Web attacks have increased rapidly in recent years. However, traditional methods are useless to track web attackers. Browser fingerprint, as a stateless tracking technique, can be used to solve this problem. Given browser fingerprint changes easily and frequently, it is easy to lose track. Therefore, we need to improve the stability of browser fingerprint by linking the new one to the previous chain.**

**In this paper, we propose LSTM model to learn the potential relationship of browser fingerprint evolution. In addition, we adjust the input feature vector to time series and construct training set to train the model. The results show that our model can construct the tracking chain perfectly well with average ownership up to 99.3%.**

***Keywords—browser fingerprint, LSTM, tracking***

## I. INTRODUCTION

In modern society, web application has become more and more complicated and powerful. It provides unprecedented convenience for people, such as online shopping, online learning, online working and entertainments, covering almost every field of our life. However, at the same time their privacy data are facing great risks as web applications are not that reliable as people think.

This work is supported by the the National Key R&D Program of China (No. 2016QY08D1602), the National Key Research and Development Program of China (NO.2016QY071405), the Key Laboratory of Network Assessment Technology at Chinese Academy of Sciences and Beijing Key Laboratory of Network security and Protection Technology.

The sensitive and valuable data is extremely attractive for attackers, which leads to web based attacks increasing. It is extremely urgent to protect the Web security. And tracking web attackers has become one of the most important challenges.

Browser fingerprint, combined by features of a user's device, network environment and web browser, can be used to identify users [1]. Browser fingerprint is widely used as a technology of web tracking, which has a wide range of purposes, including online advertising, web analytics and usability tests, and so on[2].

Originally, cookie is the first way to track web user's activity online. However, with the improvement of public awareness of privacy protection, cookies were usually deleted by user manually or by browser extensions automatically. In 2010, browser fingerprint is introduced by Eckersley as a new tracking technique, which can generate a unique identifier using attributes transferred freely by browsers. [1] Unlike cookies, browser fingerprint is without users' consent. In his paper, he showed that 83.6% of visitors in his experiment can be uniquely identified by the browser fingerprint combined by only 8 attributes. Furthermore, increasing uniqueness of the browser fingerprint has been the focus of the research in the past 7 years.

However, browser instance changes easily and frequently, which can cause the new fingerprint may be totally different from the previous one. That is to say, in order to realize the goal of tracking attackers, studies of increasing stability of browser fingerprint is as important as increasing uniqueness.

The purpose of this paper is to link browser fingerprint to one tracking chain. Recently, Antoine's team proposed two

978-1-5386-4210-8/18/$31.00 ©2018 IEEE
DOI 10.1109/DSC.2018.00038

approaches named FP-STALKER to link browser fingerprint evolution [3]. The second variant of FP-STALKER combines static rules and machine learning (random forest). In this paper, we adopt the novel thought to construct browser fingerprint tracking chain. Given browser fingerprint evolves over time, we introduce LSTM to learn the potential connection among fingerprints from one browser instance. Moreover, we construct time series based on browser fingerprint sample of different time step as input vector. In addition, we compare our method with two variants of FP-STALKER and Eckersley's algorithm. The results show that the average ownership of tracking chains can reach 0.993, which means the tracking chain can be constructed accurately.

In summary, the main contributions in this paper are:

1) Introducing LSTM to learn the potential relationship of two fingerprints originating from the same browser instance.

2) Constructing browser fingerprint time input series. We use negative sampling technology to generate adequate train set to improve the generalization ability of our model.

3) Comparing the ownership ratio of our algorithm with the results of FP-STALKER.

This paper is organized as follows. Section II describes related work. Section III gives an overview of the problem. Section IV introduces linking algorithm based on LSTM. Section V discusses the results. Finally, we conclude in Section VI.

## II. Related Work

Back to 2010, Eckersley implemented a very simple algorithm based on string comparisons to heuristically estimate whether a given fingerprint might originate from a return user. Using cookie as validation, they made correct guesses with 65% accuracy [1]. The given algorithm is simple and there's plenty of room for improving this method.

In 2015, Tomotaka *et al.* proposed a method for identifying whether plugin lists come from a same device or not based on degrees of similarity [4]. Their method used Levenshtein distance to compare two plugin lists and they showed the accuracy of the method to identify the same and different devices matched 97.94% and 97.95%.

In the same year, Liu *et al.* used 24 features and designed correlation algorithms to associate different fingerprints from a same browser instance, with accuracy increased by 24.5% than Eckersley's algorithm [5]. He used too much features that may cause the negative influence on the results.

Recently, Antoine *et al.* proved that browser fingerprint changed frequently, and presented FP-STALKER to link browser fingerprints originating from a same browser [3]. They introduced two algorithms, a rule-based one and a hybrid one that exploited machine learning. The first one was faster, while the second one's accuracy was higher. The results revealed that, on average, they could track browsers for 54.48 days, and 26 % of browsers could be tracked for more than 100 days. It is novel to use machine learning to solve the fingerprint evolve problem. We adopt following process logic in this paper.

### A. Data processing

They keep a relatively reliable fingerprint dataset filtered by the two rules as follows. [3]

1) Remove browser instances with less than 7 browser fingerprints to insure the amount is enough to track them.

2) Don't take account of fingerprints that originate from a browser instance installing anti-spoofing countermeasures. If the browser or OS changes or the attributes are inconsistent among themselves, these browser instances is discarded.

### B. Static rules

When linking browser fingerprints originating from a same browser instance, the second variant of FP-STALKER combined static rules and random forests to solve it. The rules in the hybrid algorithm originate from reasonable statistical analyses and can reduce the amount of data needed to be compared. [3]The details are as follows:

1) The OS, platform and browser family must be identical for any given browser instance.

2) The browser version remains constant or increases over time.

3) The local storage, Do not track, cookies and canvas should be constant for any given browser instance.

### C. Input feature vector

The features used to calculate the similarity of two fingerprints can be seen in TABLE I. [3]

TABLE I. Features used to calculate the similarity

| Rank | Feature | Importance |
|---|---|---|
| 1 | Number of changes | 0.350 |
| 2 | Language HTTP | 0.270 |
| 3 | User agent HTTP | 0.180 |
| 4 | Canvas | 0.090 |
| 5 | Time difference | 0.083 |
| 6 | Plugins | 0.010 |
| 7 | Fonts | 0.008 |
| 8 | Renderer | 0.004 |
| 9 | Resolution | 0.003 |

## III. Problem Overview

### A. Browser Fingerprint

Browser fingerprint is a hash value of the combination of features collected from users device, which are handed over freely by browser and don't need users' permission. Given browser fingerprint doesn't need to store any information on user's device, it is considered as a strong track technique at the beginning of its birth[1].

Browser fingerprint has been carried out in various studies, such as, tracking users[6] [7] and increasing user authentication [8] [9], which shows a possibility to track attackers. In last few years, most of the research is aimed at discovering new attributes

of browser fingerprint [10] [11] or improving the uniqueness of the browser fingerprint [12] [13]. However, since browser fingerprint changes so frequently and easily, it is not enough to track web attackers only by the uniqueness of it. There is one more thing we need to take into account is to improve stability of fingerprint.

### B. *Browser Fingerprint Changing*

The possible reasons of the change of the browser fingerprint are as follows. [3]

1) It occurs when software upgrades, for example, the upgrade of a browser may cause the change of the user agent.

2) It occurs when users carry on some operations. For instance, the list of plugins may change when a plugin is newly installed, and cookies may change when users delete it.

3) It can be affected by the behavior of the user such as the time zone may change when user carries his device during a business trip.

So few studies focus on the stability of browser fingerprint compared with uniqueness in the past several years. Considering that browser fingerprint changes so easily and frequently, research on linking fingerprints originating from a same browser instance is necessary.

### C. *Tracking Model*

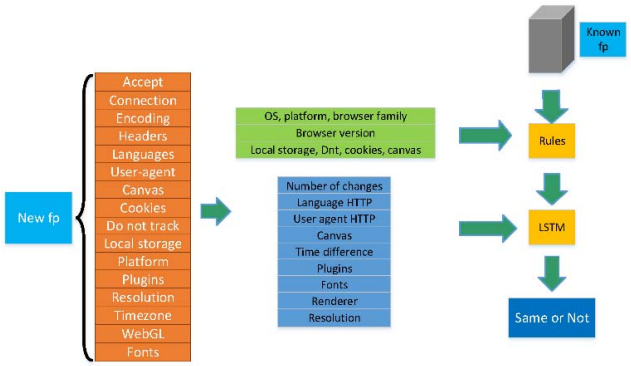


Fig. 1: Tracking model

In tracking model, there are two steps to identify if a new fingerprint belongs to a previous device. The first step is applying three static rules which are based on statistical analyses to obtain candidates. The second step is LSTM network. We train LSTM model to automatically identify browser fingerprint evolution.

## IV. Approach

When it comes to browser fingerprints linking, Antoine's team studied the evolution of browser fingerprints in FP-STALKER project. They introduced two excellent method to identify if two fingerprints originate from a same browser. [3] In this paper, instead of random forest, we introduce Long Short-Term Memory (LSTM) recurrent neural networks (RNN) to solve the problem.

### A. *Algorithm description*

In this paper, on general, we adopt the processing logic of FP-STALKER's second variant. More details can be seen in paper [3]. Considering browser fingerprints evolve over time, we introduce LSTM to link fingerprints chain. On the one hand, after applying rules on $f_u$ and all $f_k \in F$, we keep a subset of browser fingerprints $f_{ksub}$ which $f_u$ may belong to.[3] It provides us possibility to constructs time series as LSTM's input vector. F is a set of known browser fingerprints, in which we keep 3 latest fingerprints for each browser instance instead of 2 in FP-STALKER. On the other hand, we introduce LSTM to learn the potential relationship of browser fingerprint evolution. The model computes the probabilities if $f_u$ and $f_k$ come from the same browser instance.

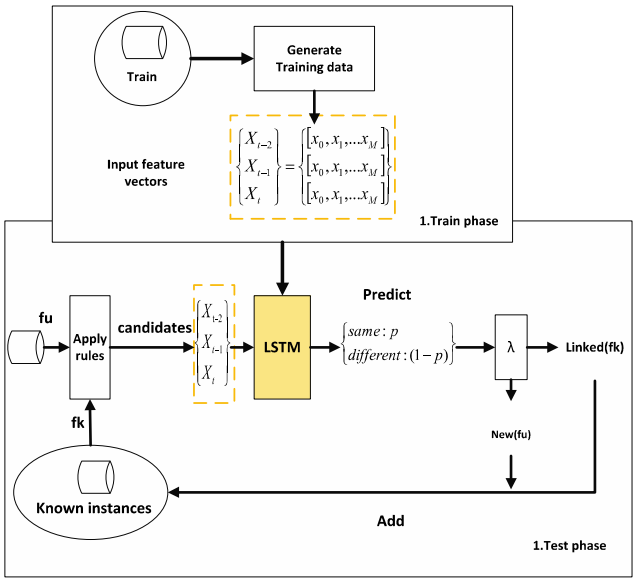


Fig. 2: Algorithm based on LSTM

### B. *LSTM as classification model*

#### *1) Input Vector*

To take advantage of the characteristics of LSTM, we transfer the single input sample to a time series. In this paper, we denote $X=[X_{t-2}, X_{t-1}, X_t]$ in current step t as input vector. Based on current time step, we use three fingerprints in last 3 time step to calculate the comparison results of all attributes between every two fingerprints. That is to say, $X=[X_{t-2}, X_{t-1}, X_t]$ $=[\text{comparison}\langle f_{t-3}, f_{t-2}\rangle, \text{comparison}\langle f_{t-2}, f_{t-1}\rangle, \text{comparison}\langle f_{t-1}, f_t\rangle]$. Comparison means that we get a single feature vector of M features $X = \langle x_1, x_2, ..., x_M \rangle$, where the feature $x_n$ is the comparison of the attribute n for two fingerprints.[3] Furthermore, considering practical reality, it is possible that the length of known fingerprint chain of target browser instance is less than 3. Therefore, we introduce zero padding strategy to deal with this circumstance (the details are presented in training LSTM part).

#### *2) LSTM model*

Since the goal of this paper is to identify a browser fingerprint comes from a previous user or a new one, it is a multi-classification problem and the number of categories is increasing over time. For the situation mentioned above, as mentioned in paper[3], it can be turned into a binary classification problem. What we need to do is to identify if the

new fingerprint and one known fingerprint originate from the same browser instance or not.

Given browser fingerprints evolve over time, it can also be considered as a time series prediction problem. Long Short-Term Memory (LSTM) recurrent neural networks (RNN), an advanced RNN architecture, is widely popular in solving sequence-based problem with long temporal dependency. Traditional RNN is mainly used for the prediction and classification of time series data. And LSTM is the most famous improvement of RNN. It can remember important information over long periods of time owing to memory cell (input gate, forget gate, output gate and cell state). LSTM is proved to be effective in time series modeling and prediction. [14] [15] Therefore, it is realistic to use LSTM to link browser fingerprint evolution.

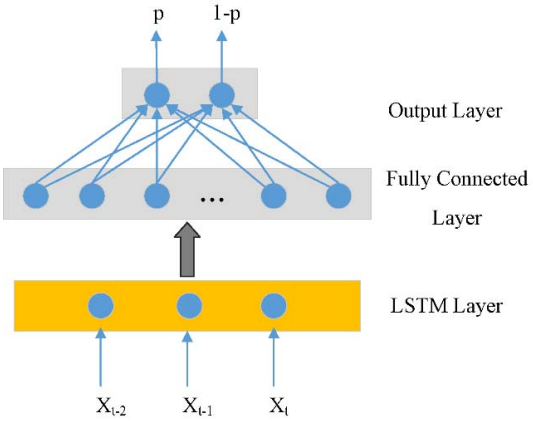


Fig. 3: Architecture of proposed LSTM model

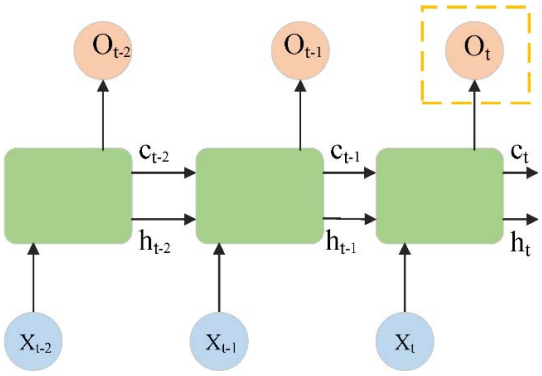


Fig. 4: Details of LSTM layer

As shown in Figure 3, our proposed LSTM model are composed of:

a) one LSTM layer with 128 hidden units. Given only $X_t$ is related to $f_u$ in input vector $[X_{t-2}, X_{t-1}, X_t]$, we use the output of the last hidden unit as the output feature vector of LSTM layer, as illustrated in Figure 2.

b) one fully connected neural layer with 96 units activated by Relu function.

c) one output layer with two neural unit with sigmoid activation. The output has 2 value of probability of the new fingerprint and the known chain belong to the same(p) and different(1-p) browser instance.

*3) Training LSTM*

When training the LSTM model, in order to make the distribution of the sample consistent with the actual situation，we constructs positive samples and negative samples as training dataset.

a) Positive samples

To construct positive samples, firstly, we separate fingerprints by browser instances(id). And then, for every browser instance, we generate fingerprint sequence with different frequency via replay mechanism in FP-STALKER [3]. Let's take browser A as an example, the generated sequence looks like $[f_{A(t0)}, f_{A(t1)}, f_{A(t2)}, \cdots f_{A(tn)}]$, $[f_{A(t0)}, f_{A(t3)}, f_{A(t6)}, \cdots]$ *et al.* Thus, for browser A, we construct positive sample as $[< f_{A(t0)}, f_{A(t1)}>, < f_{A(t1)}, f_{A(t2)}>, < f_{A(t2)}, f_{A(t3)}>]$, $[< f_{A(t1)}, f_{A(t2)}>, < f_{A(t2)}, f_{A(t3)}>, < f_{A(t3)}, f_{A(t4)}>]$, $\cdots[< f_{A(t0)}, f_{A(t3)}>, < f_{A(t3)}, f_{A(t6)}>, < f_{A(t6)}, f_{A(t9)}>]$ *et al.*

b) Negative samples

At the same time, we need to construct negative samples to train the LSTM model to identify fingerprints that belong to different browser instance. When constructing negative samples, we consider two cases.

In the first case, the known fingerprint chain is from browser A and the sample $f_u$ of current time step is from a different browser, let it be B. Then, we construct negative samples as $[< f_{A(t-3)}, f_{A(t-2)}>, < f_{A(t-2)}, f_{A(t-1)}>, < f_{A(t-1)}, f_{B(t)}>]$.

In the second case, when fingerprints of different time step belong to different browser instances, we construct negative samples as $[< f_{random(t-3)}, f_{random(t-2)}>, < f_{random(t-2)}, f_{random(t-1)}>, < f_{random(t-1)}, f_{random(t)}>]$.

c) Zero padding

In the actual production environment, the length of some known fingerprint chain may be less than 3, which is insufficient to construct time series input vector. To solve the problem, we introduce zero padding strategy.

When the length of $chain_k$ equals 1, we construct positive sample as $[zeroX, zeroX, < f_{A(t-1)}, f_{A(t)}>]$ and negative sample as $[zeroX, zeroX, < f_{A(t-1)}, f_{B(t)}>]$.

When the length of $chain_k$ equals 2, we construct positive sample as $[zeroX, < f_{A(t-2)}, f_{A(t-1)}>, < f_{A(t-1)}, f_{A(t)}>]$ and negative sample as $[zeroX, < f_{A(t-2)}, f_{A(t-1)}>, < f_{A(t-1)}, f_{B(t)}>]$, $[zeroX, < f_{random(t-2)}, f_{random(t-1)}>, < f_{random(t-1)}, f_{random(t)}>$.

After adding positive samples and negative samples to the training dataset, we use it to train LSTM model. Furthermore, we shuffle the dataset and split the dataset into training set and testing set by 8:2 to evaluate the recognition accuracy of our LSTM model.

## V. Experiments and Analysis

### A. Dataset

In this paper, we use data from dataset on Github[1] shared by Antoine's team . They collected it for 2 years through browser

[1]https://github.com/Spirals- Team/FPStalker

extensions installed by 1,905 volunteers. They shared a database of 15,000 fingerprints on GitHub. Each fingerprint had a unique identifier(id) used to identify the browser instance where the fingerprint originated from. We split the dataset into two parts, using the first 40 % of dataset as training dataset, and the last 60% as test dataset [3].

While training LSTM model, the loss and accuracy of train and test process against epochs can be seen in Figure 5 and Figure 6. Loss function measures the difference between predicted value and real value. In 38th epoch, the train loss reduced to 0.096 and the test loss reduced to 0.044. Accuracy is the proximity of measurement results to the true value. As we can see, in 38th epoch, the train accuracy reached 0.924 and the test accuracy reached 0.933. The data above showed that our LSTM model is effective to construct browser fingerprint tracking chain.

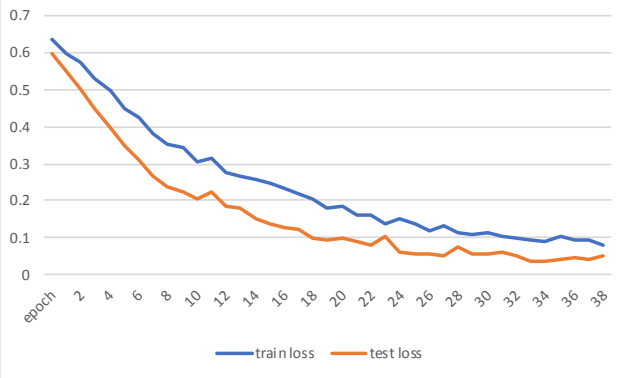


Fig. 5: BCE loss against training epochs for training and test dataset

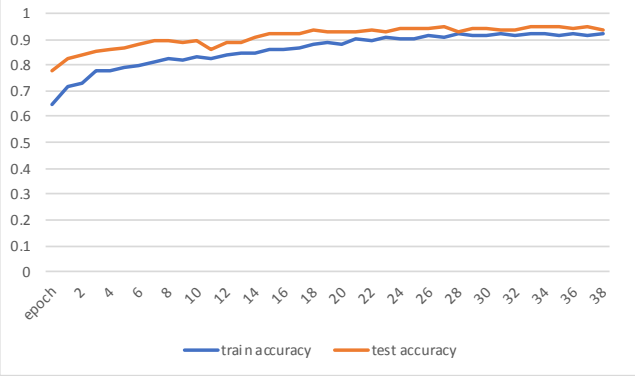


Fig. 6: Accuracy against training epochs for training and test dataset

### B. *Results and analysis*

Ownership ratio, is a performance metric proposed in FP-STALKER to evaluate the effectiveness of the algorithm. The ownership ratio equals the percentage of fingerprints belonging to chain owner. In practice, ownership ratio equals 1 means that fingerprints linked to the tracking chain originate from same browser instance.

Figure 7 shows that our LSTM model gain a better average ownership results of 0.993 than FP-STALKER's hybrid algorithm. It is significant for tracking web attackers because it requires high accuracy. And it proves the effectiveness of LSTM model to link browser fingerprint evolution.

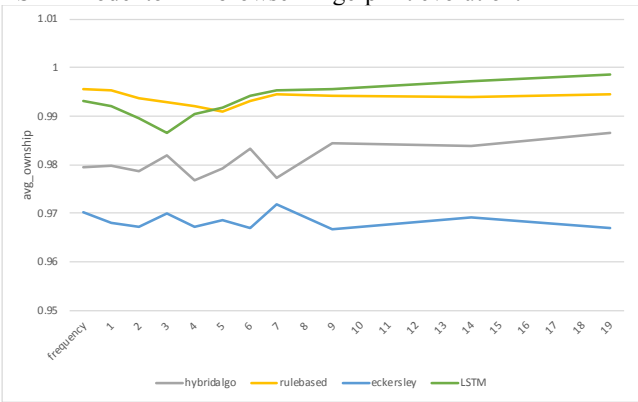


Fig. 7: Average ownership of tracking chains for the four algorithms.

## VI. Conclusion

In this paper, based on the process logic of the FP-STALKER approach, we introduced LSTM model instead of random forest to predict if two fingerprints originate from the same browser instance. Moreover, we designed the architecture of LSTM model. And by constructing browser fingerprints time series as input vector, LSTM model can learned potential connection of browser fingerprint evolution successfully. In addition, we tested our model and compared it with algorithms of FP-STALKER and Eckersley's algorithm. The average ownership ratio of LSTM model can reach 0.993, pointing out that the proposed LSTM model is effective to construct browser fingerprint tracking chain.

## References


[1] P. Eckersley, "How unique is your web browser?" Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS), 2010

[2] T. Bujlow, V. Carela-Español, J. Solé-Pareta, et al. "A Survey on Web Tracking: Mechanisms, Implications, and Defenses"[J]. Proceedings of the IEEE, 2015, PP(99):1-35.

[3] A. Vastel, P. Laperdrix, W. Rudametkin, R. Rouvoy, "FP-STALKER: Tracking Browser Fingerprint Evolutions"[C]// IEEE S&P 2018-39th IEEE Symposium on Security and Privacy. IEEE, 2018: 1-14.

[4] T. Yamada, T. Saito, K. Takasu, et al. "Robust Identification of Browser Fingerprint Comparison Using Edit Distance"[C]// International Conference on Broadband and Wireless Computing, Communication and Applications. IEEE, 2016:107-113.

[5] X. Liu, Q. Liu, X. Wang X, et al. "Fingerprinting Web Browser for Tracing Anonymous Web Attackers"[C]// IEEE International Conference on Data Science in Cyberspace. IEEE, 2017:222-229.

[6] K. Boda, A. M. Fo ́ldes, G. G. Gulya ́s, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Proceedings of the 16th Nordic Conference on Information Security Technology for Applications*, ser. NordSec'11, 2012, pp. 31–46.

[7] N. Kaur, S. Azam, K. Kannoorpatti, et al. "Browser Fingerprinting as user tracking technology"[C]// International Conference on Intelligent Systems and Control. IEEE, 2017:103-111.

[8] N. I. Daud, G. R. Haron, S. S. S. Othman. "Adaptive authentication: Implementing random canvas fingerprinting as user attributes factor"[C]// IEEE Symposium on Computer Applications & Industrial Electronics. IEEE, 2017:152-156.

[9] E. Ozan. “Password-free authentication for social networks”[C]// Computing and Communication Workshop and Conference. IEEE, 2017:1-5.

[10] D. Fifield and S. Egelman, “Fingerprinting web users through font metrics,” in *Financial Cryptography and Data Security*. Springer, 2015, pp. 107–124.

[11] O. Starov, N. Nikiforakis. “XHOUND: Quantifying the Fingerprintability of Browser Extensions”[C]// Security and Privacy. IEEE, 2017.

[12] P. Laperdrix, W. Rudametkin, and B. Baudry, “Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints,” in *37th IEEE Symposium on Security and Privacy (S&P 2016)*, 2016.

[13] K. Mowery and H. Shacham, “Pixel Perfect : Fingerprinting Canvas in HTML5,” Web 2.0 Security & Privacy 20 (W2SP), pp. 1–12, 2012.

[14] M. A. Zaytar and C. El Amrani, “Sequence to sequence weather forecasting with long short-term memory recurrent neural networks,” Int. J. of Computer Applications, vol. 143, no. 11, 2016.

[15] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, “An overview and comparative analysis of recurrent neural networks for short term load forecasting,” arXiv preprint arXiv:1705.04378, 2017.