An Adaptive Authentication Based on Reinforcement Learning

Ziqi Cui[1], Yongxiang Zhao[1], Chunxi Li[1], Qi Zuo[2], Haipeng Zhang[2]
1. Information and electronic school, Beijing Jiaotong University
2. Beijing Cloud Computing Key Technique and Application Key Laboratory, Beijing Computing

***Abstract*—Different authentication confidence probabilities are required when doing different amount of online commercial transfer. We should use different combinations of authentication methods according to different level of authentication confidence requirements. This paper proposes a new multi-factor authentication based on reinforcement learning to realize adaptive authentication. Numerical results show our scheme can reduce the authentication cost while satisfying authentication confidence.**

I. INTRODUCTION

Alipay and Wechat are widely used when doing shopping and transferring money online in daily life nowadays. The major concern of such online money transfer applications is to authenticate the user of the applications. According to the number of factors used, authentication methods can be classified as mono-factor method and multi-factor method. Mono-factor system can be further classified into following categories [1]: something you are, e.g., fingerprint and voice recognition; something you do, e.g., gesture based authentication; something you know, e.g., password authentication; and something you possess, e.g., USB key fob. Multi-factor systems combine information from multiple factors to overcome limitations such as non-universality, noisy sensor data, and large intra-user variations commonly encountered in mono systems.

All above authentication methods use the same authentication strategy, which means the number of authentication method and their combination are fixed. Thus, the confidence probability, representing the authentication outcome's confidence extent, is also fixed. However, in real world, different amount of online money transfer requires different level of confidence probability. For example, confidence probability of 0.9 is enough when we transfer 10 dollars, but it is obviously not sufficient when the transfer amount is increased to 1 million dollars. This inspires us that we should select authentication method according to the requirement of confidence probability and the prior knowledge of users. To this end, we need to solve the problem of choosing authentication method according to the authentication history when facing multiple optional mono-factor methods and different confidence probability requirements.

In this paper, we apply reinforcement learning to form the authentication selection policy, which can adaptively select authentication test until the confidential probability requirement is satisfied.

This work was supported in part by National Natural Science Foundation of China under Grants 61572071 and 61872031.

II. ADAPTIVE MULTI-FACTOR AUTHENTICATION

A. Application scenario

As shown in Fig.1, the system consists of a server and two types of users: genuine (T) and impostor (F), which are legal and illegal customers of the system respectively. When a user accesses to the server, the server needs to identify the real type of the user by requiring the user to conduct a series of tests. Our goal is to find an efficient authentication strategy for the server to figure out the type of users, which should satisfy the required confidence probability and spend less authentication time as much as possible.

We assume there are N sequentially numbered Mono-factor tests. We use symbol p^m_{ij} to represent an outcome probability, meaning that we use test m for user type i to get an output j (pass or not pass the test). We denote authentication time of test m as C_m. Both p^m_{ij} and C_m are m^{th} statistical data known by the server.

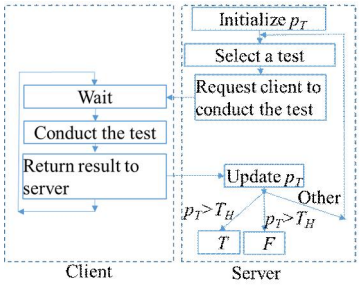

Fig.1 System and flow chart

B. Flow chart of client and server

As shown in Fig.1, on the client side, once receiving a test request from the server, client conducts the test and then returns the test result to the server.

Two symbols, p_T and p_F are to describe users' state in the server, with p_T and p_F representing the probability that the server believes the user is genuine(T) and impostor(F) respectively, where $p_T+p_F=1$. In this paper, we uniformly quantify p_T and p_F as discrete values with step $1/K$ (K=1000), thus value of p_T and p_F come with the range of [0, $1/K$, $2/K$,…, $(k-1)/k$, 1]. The server also keeps a $(K+1)\times 2$ strategy table T_p, whose 1st and 2nd columns save the $(K+1)$ values of p_T, and the number of the tests chosen corresponding to the 1st column respectively.

In Fig.1, sever works as following: The server initializes p_T

978-1-7281-2279-2/19/$31.00 ©2019 IEEE

according to certain prior probability and then enters the following loop: First, it retrieves the test number from table T_p according to the user's current value of p_T and request the client to conduct the test. Second, the server will update value of p_T based on the test result returned by the client, where the details will be introduced in next sub-section. Finally, the server will make a decision: if $p_T > T_H$ or $p_T < T_L$, the server will break the loop and output T or F respectively, otherwise, the server will loop the above steps. Here, T_H and T_L are confidence threshold for server to determine users' type.

C. Updating p_T and p_F

We use symbols S and F to represent the test result that user has passed and failed the test respectively. When the server gets a S result from client, then we have following equations according to Bayes formula.

$$p(T/S) = p_T \times p^m_{TS} / P(S). \quad (1)$$

$$p(F/S) = p_F \times p^m_{FS} / P(S). \quad (2)$$

$$p(S) = p_T \times p^m_{TS} + p_F \times p^m_{FS}. \quad (3)$$

Where, $p(T/S)$ and $p(F/S)$ is the probability that a user is T or F respectively on condition that the test is successful. $p(S)$ represent probability that the test is successful. We set p_T and p_F to $p(T/S)$ and $p(F/S)$ respectively. When getting a F result, we can use the same method to update p_T and p_F.

D. Creating Policy Table T_p

As shown above, the key to enable server to select test wisely is to create Policy Table T_p legitimately. We will use reinforcement learning to realize this function. First, we define the state s in reinforcement learning as the value of p_T and action set at each state is set of tests. Then we define reward function in reinforcement learning [2] as follows.

$$r(s'|s,a) = |s' - (T_H - T_L)/2|. \quad (4)$$

$r(s'|s,a)$ represents reward obtained when following events happen: users' current state is s, client conduct test a, and its state will update to s' by steps shown in subsection II.C. By this function, the closer the state s' approaches to T_H or T_L, the larger the reward is.

Furthermore, we name the states set $\{s|\ s>T_H$ or $s<T_L\}$ and $\{s|\ T_L<=s<=T_H\}$ as known set and unknown set respectively. We add an extra condition in reward function to increase the importance of s which belongs to the known set: if both s and s' belongs to the known set, $r(s'|s,a)$ will return with a large number(it is 1000 in this paper); if s belongs to the known set while s' does not, $r(s'|s,a)$ will return with a rather small number(it is 1 in this paper). By this way, the state in the known set gains more importance and algorithm will prefer to select the test to enter known set quickly.

Another input parameter in reinforcement learning is $p(s'|s,a)$, which represents the probability that state s transfers into state s' by taking action a. First, given a state s and an action a, and assuming the test is successful, we can find the value p_T that will transfer and its corresponding transferring probability by equation (1) and (3). Then, we repeat the similar steps by assuming the test fails. Finally, we set $p(s'|s,a)$ to be zero except the above two cases.

After defining $r(s'|s,a)$ and $p(s'|s,a)$, we can use average reward semi-markov decision process and its corresponding code in [2] to produce table T_p.

III. Numerical Results

A. Baseline schemes

We use following metrics to evaluate the performance authentication progress: false positive which represents the probability that a F user is authenticated as a T user, true positive which represents the probability that T user is authenticated as a T user, and average cost represents average cost to authenticate a user.

We use two baseline schemes to compare with our scheme shown in the above section. Both schemes are the same with our scheme shown in Fig.1 except that they use different method to choose test at step 2 in the server side: the one chooses a test randomly; the other fixes to one test and output the smallest average cost of all the tests'.

B. Experiments setting

The following table shows the setting of five types of test:

Table 1 Parameters settings of tests

Test type	p^m_{TS}	p^m_{FS}	C_m
1	0.3	0.2	100
2	0.35	0.3	250
3	0.55	0.45	400
4	0.75	0.60	550
5	0.90	0.1	1000

We have 100,000 users to be tested, and 60% of users belong to T and the rest belong to F. T_H and T_L are set to be 0.95 and 0.05 respectively.

C. Test results

Simulation results are shown in Table 2.

Table 2 Simulation results

Schemes	false positive	true positive	average cost
Baseline 1	0.0415	0.9747	4509.8
Baseline 2	0.011	0.9716	2437.5
Our scheme	0.011	0.9886	2060.8

As is shown above, our scheme can save about 54% and 15% cost than baseline 1 and baseline 2 respectively while obtaining required confidence probability. We have also tried other tests' parameter settings and found that average cost of our scheme is always no more than the average cost of either baseline 1 and 2.

Reference

[1] N.Gunson, D.Marshall, H.Morton, M.Jack, User perceptions of security and usability of single-factor and two-factor authentication in automated telephone banking, Computers & Security,30(2011),PP:208-220

[2] A. Gosavi. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning, Springer, New York, NY, Second edition, 2014, http://web.mst.edu/~gosavia/matlab-codes.zip