PAS: Predicate-based Authentication Services Against Powerful Passive Adversaries

Xiaole Bai† , Wenjun Gu† , Sriram Chellappan‡, Xun Wang+, Dong Xuan† and Bin Ma⋆
† Dept. of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{baixia, gu, xuan}@cse.ohio-stat.edu
‡ Dept. of Computer Science
Missouri Univ. of Sci. and Tech.
Rolla, MO 65409
chellaps@mst.edu
+ Cisco System Inc.
San Jose, CA 95134
xunwan@cisco.com
⋆ David R. Cheriton School of Computer Science
University of Waterloo
ON, Canada, N2L 3G1
binma@cs.uwaterloo.ca

Abstract

Securely authenticating a human user without assistance from any auxiliary device in the presence of powerful passive adversaries is an important and challenging problem. Passive adversaries are those that can passively monitor, intercept, and analyze every part of the authentication procedure, except for an initial secret shared between the user and the server. In this paper, we propose a new secure authentication scheme called Predicate-based Authentication Service (PAS). In this scheme, for the first time, the concept of a predicate is introduced for authentication. We conduct analysis on the proposed scheme and implement its prototype system. Our analytical data and experimental data illustrate that the PAS scheme can simultaneously achieve a desired level of security and user friendliness.

1 Introduction

Authenticating the identity of a human user to a server is critical in security. While it is reasonable to assume that secrets initially shared between a user and a server were done so securely, subsequent authentication may be conducted in malicious environments, which could expose secrets to adversaries for malicious impersonation later on. Some typical instances have been reported in [5], [6] and [7]. The significance of this issue arises from the recent dual trends of increasingly mobile users and rapidly-advancing hardware and software technologies that the malicious can exploit.

One natural approach that can be used in untrustworthy environments is one-time passwords. However, they are generally difficult to memorize. Users may have to store them in or generate them with some auxiliary device (*e.g.*, cell phones or one-time password generators) that is vulnerable to theft. In some cases, one-time passwords are provided in a letter and sent to users, *e.g.*, Nordea Bank in Finland. Then users may also have to refer to the letter when typing the passwords during authentication process, which is both cumbersome and vulnerable to observation with hidden cameras. Thus the development of a secure authentication scheme that the general public can easily use is urgently needed.

1.1 Problem Statement

In this paper, we address the issue of how a human user can securely authenticate himself to a server in the presence of powerful passive adversaries *without* assistance from any auxiliary device.

Attack model: powerful passive adversaries. By *powerful*, we mean adversaries that can monitor, intercept and analyze each part of the authentication procedure (*i.e.*, all the inputs and outputs of the authentication process), except for an initial secret pre-shared between the user and the server. By *passive*, we consider adversaries that do not disrupt the authentication process or change or create new content transferred during the process. The common goal of these adversaries is to subsequently impersonate the valid user later on. We comment that this attack model is stronger than "shoulder surfing". Concealed input, *e.g.*, fingerprints, can successfully defend against shoulder surfing, but there is no strong defense against our attack model, where the fingerprint could be recorded.

User constraint: no auxiliary devices. In this paper, we focus on the design of authentication procedures that can be accomplished by most human users without help from any auxiliary device. This constraint is important in a practical sense, and also raises significant design challenges. We notice there are many interesting works in which users authenticate themselves with the assistance of extra devices,

e.g., [3, 14–16, 18–20]. The major problems with these devices are production and distribution costs, sensitivity to theft and loss, and the inconvenience of carrying them.

Design goal: secure authentication. By *secure authentication*, our objective is to protect users from malicious impersonation after the authentication procedure has been successfully accomplished in the presence of powerful passive adversaries. Clearly, authentication methods based on a fixed input as the password for every login attempt cannot achieve this.

1.2 Challenges

This problem's essential difficulty lies in the user constraint, *i.e.*, the authentication procedure should be handled without any help from auxiliary devices.

To overcome powerful passive adversaries, the well known zero-knowledge proof-based approaches are natural candidates. However, zero-knowledge techniques are based on mathematically hard problems [1]. Since humans' "computational power" is much less than that of machines, these approaches are simply too difficult for a user to handle. The fact that zero knowledge techniques are hard to be applied here is critical. It implies that information leakage in each authentication session can be feasibly taken advantage of in our problem setting. Let the entropy of the pre-shared secret space be denoted by $\mathcal{H}$, and the entropy of information leakage resulting in secret space shrinking in each session be denoted by $\Delta\mathcal{H}$; inevitably $\Delta\mathcal{H} > 0$. After being used for $\lceil \mathcal{H}/\Delta\mathcal{H} \rceil$ times, the pre-shared secret will be revealed. This upper bound of the number of authentication sessions reveals two facts. First, the pre-shared secret cannot be used forever since $\Delta\mathcal{H} > 0$. That is, the secret has to be renewed after being used for a certain number of authentication sessions. Second, if we want to increase the number of authentication sessions of a pre-shared secret, we must either decrease the information leakage or increase the secret space, or both.

The first approach, decreasing the information leakage, is difficult. On the one hand, operations needed to finish an authentication session are limited when users' computational power is constrained. Mathematical operations, *e.g.*, modular arithmetic, are difficult for many people, especially when very large numbers are involved. On the other hand, powerful *satisfiability solvers* (*SAT solvers*) can easily exploit information leakage. Formally, SAT solvers are logical cryptanalysis tools [13]. The attacker first encodes the authentication scheme as an SAT problem and then uses state-of-the-art SAT solvers to obtain all the possible secrets that satisfy the responses observed. Note that it is even feasible to encode the U.S. Data Encryption Standard (DES) as a SAT problem [12], and state-of-the-art SAT solvers are powerful enough to handle problems with hundreds of thousands of variables efficiently [21]. Following this approach, to date there is no secure authentication scheme that the general public can easily use.

The second approach, increasing the secret space, is also difficult. We must then address the challenge of limited human memory capacity and the desired huge secret space. It is challenging for humans to memorize a secret with high entropy, and consequently, very little work follows this direction. To overcome this challenge, existing work requires extensive training that lasts at least a few days, or even months. It is nontrivial to design an authentication scheme that does *not* require extensive training time.

1.3 State Of The Art

Although the problem has been studied for over ten years, to date existing solutions are far from usable. There is some pioneering and inspiring work that aims to decrease information leakage. Matsumoto and Imai in [10], Wang *et. al.* in [25] , and Matsumoto in [11] proposed solutions based on vector computation. Hopper *et. al.* in [4] proposed a scheme based on a conjectured hard problem—learning parity in the presence of noise. One major downside of this work is that the required user computation is too difficult to perform without an auxiliary device. For instance, it is reported only 10% of users can handle the approach in [4]. In an attempt to increase the secret space, in [23], Weinshall *et. al.* propose a scheme in which a user selects between 100 and 200 pictures from a set of 20,000 pictures as the one-time secret that is recognized for subsequent authentication sessions. However, this approach requires approximately three months of training time to remember such a large amount of information. In 2006, Weinshall proposed an interesting scheme in [24]. In this solution, a user memorizes 30 pictures out of 80 pictures as the secret. During one round for an authentication session, the server randomly displays some pictures on the screen in a tabular form. According to the positions of the pictures on the table, the user follows a rigorous protocol to discover a number, which is the password. The process continues over multiple rounds before the user is successfully authenticated. Two days of training is required to memorize the secret pictures and get familiar with the authentication procedure. In a subsequent work [2], it was shown that under SAT attack, the secret memorized in [24] has to be renewed after around six authentication sessions.

It is worthwhile to note that the schemes proposed in [8], [17], [26] and [27] against shoulder surfing attack can also be used against our attack model. However, these schemes overlooked SAT attacks in their designs.

1.4 Our Contributions

We design a new authentication scheme called *Predicate-based Authentication Services* (PAS) following the second aforementioned approach—increasing the secret space—to delay disclosure of the secret.

PAS balances security and user-friendliness by introducing a new element into the authentication scheme design, namely, predicates. A *predicate* typically is a verb phrase

template that can be defined: (a) as a property of certain objects, or (b) as a relationship among objects, or (c) as a value quantifying certain properties of the object as represented by the variables in the predicate. In PAS, the objects are characters, and the predicates encompass all three definitions.

In the paper, we show our PAS scheme can achieve a desired level of security against powerful passive adversaries. We also validate the user-friendliness and system-friendliness of our PAS scheme by building a prototype system and carrying out real user experiments. The experimental results show that PAS can be handled by *all* recruited users. And it requires *no* specific training time and takes a user less than 10 minutes on average to learn and prepare. This demonstrates PAS has much greater usability compared with existing solutions under the same problem setting. The desired features of PAS make it a candidate for a human authentication scheme that can be rapidly adopted against powerful passive adversaries for use in untrustworthy environments.

The rest of our paper is organized as follows. Section 2 presents PAS working procedure and its design details. Section 3 presents an extended PAS scheme. Sections 4 presents theoretical analysis and numeric results. Section 5 presents our prototype implementation and usability validation, followed by conclusions in Section 6.

2 Predicate-based Authentication Service

In this section, we first discuss the key concepts in the PAS scheme: secrets and predicates. We then illustrate the whole authentication process via an example, followed by a detailed design of the PAS scheme.

2.1 Secrets and Predicates

In the PAS scheme, the *secrets* are what a prover P (*i.e.*, a user) shares with a verifier V (*i.e.*, the PAS server) during the registration stage. In particular, P shares two secrets with V. Each secret consists of two parts: an integer that will act as the *cell index* and a string called the *secret word*. For example, two secrets could be "23 sente" and "41 logig".

In the PAS scheme, *predicates* are generated from the secrets. Before P attempts to login, V will prompt for an integer I asking P to use the I^{th} character in the two secret words to generate two predicates. Suppose $I = 15$ and the secrets are the same as those above. P can construct two predicates, that are, "23e" and "41g", in which "e" is the 15^{th} character in the first secret word "sente" (characters in words are counted in a "looped" way) and "g" is the 15^{th} character in the second secret word "logig". The interpretations of these two predicates are "there is a character 'e' in cell (2, 3)" and "there is a character 'e' in cell (4, 1)", respectively. To sum up, the secrets are what P memorizes initially. For each login session, P generates two predicates

Figure 1. Two challenge tables in a round (shrunk screenshot).

based on the value of I and the secrets, and then use them to login. The login process is detailed in the following section.

The key difference between the PAS scheme and traditional password-based schemes lies in the separation between secrets and predicates. In traditional password-based schemes, there is no concept of predicates, and *the secrets themselves* are used directly in the login process. Clearly, such schemes are vulnerable under powerful passive adversaries. In PAS, the predicates derived from the secrets are further hidden (detailed mechanisms will be presented later) and can vary from session to session.

2.2 A Login Example

In this section, we will give an example to illustrate the whole login process. We use the same example above, in which two secrets are "23 sente" and "41 logig". After being given $I = 15$ by V at the start of the login session, P constructs two predicates "23e" and "41g". Authentication in PAS takes several rounds. In the following, we discuss the process in one particular authentication round.

For each round, V presents two challenge tables, as shown in Fig. 1, and one response table, as shown in Fig. 2. P first checks if the first predicate "23e" holds or not in two challenge tables. Noticing there is a character "e" in cell (2,3) of both challenge tables (case insensitive), P memorizes "Yes Yes" for the first predicate. Then, P conducts the same lookup process for the second predicate "41g". Since there is a character "g" in cell (4,1) of the first challenge table, but there is no character "g" in cell (4,1) of the second challenge table, P gets "Yes No" for the second predicate. Based on what P memorizes above ("Yes Yes" and "Yes No"), the response can be obtained in the response table with "Yes Yes" as the row index and "Yes No" as the column index, which is "RM" in this example. After typing the response, the user is led to the next round.

Such process is repeated for each round. In the different rounds of a session, the two predicates used by P are not changed while both challenge tables and response table provided by V will change. P can successfully login only if P has provided the correct responses for all the rounds in

Figure 2. A response table (shrunk screenshot).

the session.

2.3 PAS Design Details

2.3.1 Challenge Table Design

In each round of an authentication session, ℓ challenge tables are generated by V and sent to P. We first show how these challenge tables are constructed, and then show how the challenge table number ℓ is decided.

All ℓ challenge tables are of the same size with m rows and n columns. Thus, there are total $M = m \times n$ cells in each challenge table. Each cell in the tables is filled with a certain number of characters. Let H denote the set of all possible characters that can be put into the cell. Let H' denote the set of characters contained in one cell ($H' \subseteq H$). All characters in H' are unique and randomly chosen from H. We denote the maximum value of $|H'|$ ($|A|$ denotes the cardinality of a set A) by τ. The value of τ, *i.e.*, the maximum number of characters shown in each cell, is decided by

$$(\tau - 1)/|H| < 0.5 \leq \tau/|H|. \quad (1)$$

In the above equation, the left-hand side is the probability that one predicate evaluates to TRUE when there are $\tau - 1$ distinct characters in each cell. Similarly, the right-hand side is the probability that such a predicate evaluates to TRUE when there are τ distinct characters in each cell. For each cell, V will fill τ distinct characters in it with certain probability β, or fill $\tau - 1$ distinct characters in it with certain probability $1 - \beta$. Using τ and $|H|$, the value of β can be obtained from the following equation

$$\beta\tau/|H| + (1 - \beta)(\tau - 1)/|H| = 0.5. \quad (2)$$

Such a value of β can guarantee that a predicate evaluates to TRUE with probability 0.5. By manipulating the above equation, we have

$$\beta = |H|/2 - \tau + 1. \quad (3)$$

The value of β will not change after having been computed as long as the values of τ and $|H|$ are not changed. When generating the challenge tables, V first generates a random number r for each cell, which is uniformly distributed over the range $[0, 1]$. If $r < \beta$, V randomly selects τ different characters from H and puts them into the cell. Otherwise, V selects $\tau - 1$ characters. Each cell is filled independently. After filling all cells in all ℓ tables, V sends these tables to P. In the example shown in Fig. 1, the set of all possible symbols is $H = \{A, \cdots, Z\}$ with $|H| = 26$. We then have $\tau = 13$ and then from (3) we have $\beta = 1$. That is, there are always 13 different characters filled in each cell. We comment that challenge tables can be generated by incorporating the CAPTCHA technology to defeat attackers' bots and force attackers to extensively involve human beings for visually interpreting them.

Note that the construction approach described above does not take any content information of predicates as input. This allows the challenge table construction to be independent of the user identity and his secret shared with V. Thus, the attacker cannot differentiate the secrets of two users by observing their challenge tables. Furthermore, the challenge tables constructed by this way have a property stated as follows. Let the predicate used for each session define a random variable S, and let the result from table lookup define a random variable A. Then we have the following Theorem 2.1, which states that, even the adversary knows the answer of a table lookup, he is not able to tell whether this answer is obtained by a random guess or by applying any predicate. We omit its proof due to space limitations.

Theorem 2.1 *In the PAS scheme, for any predicate s,*

$$Pr\{A = Yes | S = s\} = Pr\{A = Yes\} = 1/2. \quad (4)$$

We now introduce references that can help decide the value of ℓ, the number of challenge tables shown per round. The value of ℓ is decided by the capacity of human visual short term memory. When there are p predicates, then each table will generate p answers, each of which is a "Yes" or "No". Consider each such answer as a symbol. There are totally $p\ell$ symbols that should be memorized by P in one round before looking up the response table. In psychology, Miller [9] shows that people's average short-term memory capacity is 5–9 symbols. By observing neural activity, Vogel and Machizawa in [22] show in 2004 that the effective short term memory can be around 4 symbols for a average people. These studies are valuable when we design our PAS system to achieve a good usability. In our prototype system, we use $p = \ell = 2$.

2.3.2 Response Table Design

In each round, one response table is generated by V and sent to P. Each cell in the table, containing a typeable response, is indexed by sequences of $b_1 b_2 \ldots b_\ell$, where each b_i is a "Yes" or "No" and ℓ is the number of challenge tables in each round. The assignment of "Yes" or "No" for b_i indicates whether or not one predicate is satisfied in challenge table i ($1 \leq i \leq \ell$). If there are p pre-shared secrets, *i.e.*, p predicates will be generated, then this table will be of p dimensions. That is, each cell, and thus the response, will be indexed by p sequences. Fig. 2 shows an example where $p = 2$. We comment that showing a table with more than two dimensions is achievable, *e.g.*, a Flash file

can show a three-dimensional table by automatically showing two-dimensional subtables in it.

The response table is carefully designed to prevent possible information leakage. First, it prevents adversaries from obtaining the exact value of $b_1 b_2 \dots b_\ell$ after having observed the input responses. In PAS, there are $2^{p\ell}$ cells in one response table. The table is filled by using only 2^ℓ responses. It implies that there are $2^{(p-1)\ell}$ possible ways to index one single response in PAS. Thus, the reverse mapping, *i.e.*, from a response to sequences of $b_1 b_2 \dots b_\ell$, will have multiple choices. Second, it prevents frequency skew of appearance of each response. In the PAS scheme's current design, each response appears once and only once in each dimension. For example, when $p = 2$ as shown in Fig. 2, the verifier V fills 2^ℓ different strings into this table in such a way that the same string does not appear in any single row or column twice. We state our uniformness property in the following theorem. Suppose there is a probability distribution on the response space $\mathcal{R}$. Thus the response defines a random variable R. We denote the probability that the response is r by $Pr\{R = r\}$. Similarly, we assume the predicates used for one authentication session, may follow certain probability distribution, which is decided by the pre-shared secret between P and V. Let the predicate also define a random variable S. We denote the probability that predicate s is chosen by $Pr\{S = s\}$.

Theorem 2.2 *In the PAS scheme, for any response $r \in \mathcal{R}$ and any predicate s,*

$$Pr\{R = r|S = s\} = Pr\{R = r\} = 1/|\mathcal{R}|. \qquad (5)$$

We omit its proof due to space limitations. In Theorem 2.2, the first part of the equation states that the probability that r is the response, given any s as the predicate, is equal to the probability that r is the response. That is, the probability distribution of the response is independent of the chosen secrets/predicates. Thus, the attacker cannot infer any information about the secret of prover P by *only* observing the probability distribution of P's responses. The second equation in Theorem 2.2 states that every possible response occurs with the same probability in PAS. Therefore, there is no skew in the response distribution that adversaries can exploit. Third, the response table prevents them from taking advantage of correlation of response locations in the response table. Correlation of response locations can result in attacks with low computational complexity. In the PAS scheme, the locations of responses in the response table are randomly generated for each round such that no fixed correlation can be obtained over rounds by adversaries. Technically, it can be done by two steps. In the first step, let each cell with the same sum of p indexes have the same response. In the second step, randomly shuffle this p dimension-wise. For example, when $p = 2$, we have a two dimensional response table. This table can be generated by first filling the same response into cells with the same value of $(i + j) \bmod 2^\ell$ where i and j are row and column indices respectively, and then shuffling the columns and rows randomly. Finally, the response table prevents the random guessing attack in the response field for each session. The probability for successful random guess is $1/2^{\ell n_r}$ that should be small enough where n_r is the number of rounds in one session.

Response tables are generated by incorporating the CAPTCHA technology. Any type of CAPTCHA, *e.g.*, character recognition based or image recognition based, can be adopted as long as the corresponding responses are typeable.

2.3.3 Secret and Predicate Design

The predicate in the PAS scheme takes form of (u, v, h), where u and v decide the cell position and h denotes a character. The values of u and v are decided by the size of the challenge tables. To decide the value of character h, in each authentication session, one index I will be generated by V and sent to P, indicating which character in the secret words will be used to construct the predicates. Each predicate index I can only be used for a limited number of sessions that end up with "login successful," due to inevitable entropy decrement. The value of this number, denoted by t_{max}, is decided by a variety of system parameters. We defer the expression of t_{max} to Section 4.

In the PAS scheme, the predicate is interpreted as "there is one character h in a cell indexed by row u and column v." This interpretation is based on the property of a cell with *fixed* location. Compared to other options that may involve relative positions like "there are two neighboring cells containing character A and B respectively," the current design is preferred. It provides the convenience for users to check the satisfaction of predicates efficiently. Instead of looking though all the cells in the challenge table, users now can quickly tell the results after they get familiar with the locations of cells they remember.

System friendliness is also considered in our current PAS predicate design, where the objects chosen are based on characters. Although general multimedia objects are possible, the characters are generally more light-weight in terms of manipulation, storage and delivery.

In our previous example, the secret word memorized is "sente" and "logig". In fact, a secret word can be constructed in any way as long as the user is comfortable memorizing it without compromising its security. There are plenty of materials online teaching people how to choose a good password that satisfies the long term memory and security. One possible way is to create new words that can be easily memorized. For example, the secret word "sentenceyz", which has 10 characters, is constructed by concatenating an English word "sentence" with a short character sequence "yz". Another possible way to construct a secret word is by interleaving two easily memorized character sequences. For example, the secret word "laobgcidce", which also has 10

characters, is constructed by interleaving an English word "logic" and a short character sequence "abcde".

3 The Extended PAS

In the above section, we illustrated the whole login process in our PAS system via an example, and then presented the design details for each component in PAS. We comment that, the various facets of the PAS scheme could be modified to meet the different demands on security and/ or user-friendliness. In this section, we briefly introduce the extended PAS scheme.

The most important extension is on predicate construction. In our previous example, each predicate has an atomic term like "there is a particular character in a certain cell". In an extended PAS scheme, a predicate can contain multiple atomic terms connected with ORs ("$\vee$"s). Let the number of terms be denoted by k. A generalized predicate can take the form "$(u_1, v_1, h_1) \vee (u_2, v_2, h_2) \vee \cdots \vee (u_k, v_k, h_k)$", where u_i and v_i $(1 \leq i \leq k)$ are cell indices and h_i $(1 \leq i \leq k)$ is a character. A generalized secret then takes the form as "$u_1, v_1, u_2, v_2 \ldots, u_k, v_k, S[1]S[2] \ldots S[len]$", where $S[i]$ $(1 \leq i \leq len)$ denotes the i^{th} character in the secret word. At the start of every session, k predicate indices, I_i $(1 \leq i \leq k)$, will be generated by V and sent to P. For example, P memorizes "2345 sente", where $u_1 = 2, v_1 = 3, u_2 = 4, v_2 = 5$ and $len = 5$. At the start of one session, two predicate indices $I_1 = 1$ and $I_2 = 2$ are generated. P then constructs a predicate as "(23s) $\vee$ (45e)", and then uses this predicate to check the challenge tables. Note the number of terms in each predicate k can be adjusted by different users with different memory capabilities.

In the extended PAS scheme, although we can replace any of the ORs in the above predicate by ANDs ("$\wedge$"s) and add any number of parentheses necessary to generate other valid predicates, we recommend using ORs as above. The reason is that using ORs can minimize the number of characters shown in each cell. This makes our scheme more user-friendly, since the number of characters shown on-screen will decrease, as will be the time spent on authentication. This is summarized in the following theorem. Its proof is omitted due to space limitations.

Theorem 3.1 *In the extended PAS scheme, τ is minimal when all the logical connectives in the predicate are ORs.*

In the extended PAS scheme, k predicate indices will be generated by V and sent to P in each authentication session. They indicate the characters in the secret words that will be used to construct the predicates. These k indices are denoted by I_i $(1 \leq i \leq k)$. To generate I_i, two requirements must be satisfied. First, all k indices must be distinct for one round. This requirement is to avoid unnecessarily shrinking the predicate size for each session. Second, each predicate index, *i.e.*, I_i, can only be used for a limited number of sessions that end up with "login successful", due to inevitable entropy decrease. The expression of this number, denoted by t_{max}, will be provided in Section 4. When designing the challenge tables for the extended PAS scheme, we decide the value of τ following

$$1 - \left(1 - \frac{\tau - 1}{|H|}\right)^k < 0.5 \leq 1 - \left(1 - \frac{\tau}{|H|}\right)^k. \quad (6)$$

Using τ, k and $|H|$, the threshold value β can be obtained in the V side from the following equation:

$$1 - \left(1 - \left(\beta \frac{\tau}{|H|} + (1 - \beta)\frac{\tau - 1}{|H|}\right)\right)^k = 0.5. \quad (7)$$

Such a value of β can guarantee that a predicate evaluates to TRUE with probability 0.5. By reforming the above equation, we have,

$$\beta = |H| \left(1 - \left(\frac{1}{2}\right)^{\frac{1}{k}}\right) - \tau + 1. \quad (8)$$

When $k = 1$, equations (6), (7) and (8) are the same as equations (1), (2) and (3), respectively. Finally, we note that Theorems 2.1 and Theorem 2.2 still hold for the extended PAS scheme.

4 Security Analysis

4.1 Preliminaries

We consider three types of attacks: *brute force attack*, *random guessing attack* and *SAT attack*. These attacks may have different attack targets. There are three types of attack targets in PAS: *secret*, *predicates* and *responses*. The definitions of attacks and their relationship with the attack targets are detailed below.

– *Brute force attack:* The attacker first obtains a set of all possible secrets, and then tries them one by one. Such an attack does not take predicates or responses as attack target, since they vary in different rounds.

– *Random guessing attack:* This attack randomly tries one possible secret, predicate or response in an authentication session. All three types of attack targets apply here.

– *SAT attack:* As mentioned in Section 1, SAT attack is the main threat that efficiently takes advantage of information leakage to reveal the secret. Hence, in our analysis, we assume that the SAT attacker can perfectly use knowledge from previously-observed authentication sessions. In PAS, only two types of attack targets, namely secrets and predicates, apply to this attack.

In order to evaluate the security and user-friendliness of our PAS scheme, we introduce two metrics here.

– *Cardinality of attack set:* This metric is denoted by $|\mathcal{S}|$. $\mathcal{S}$ consists of all possible values of a target. It can be a set of secrets, predicates or responses. There are no leads to further narrow down $\mathcal{S}$ except for trying possible responses.

High $|\mathcal{S}|$ is desired. $|\mathcal{S}| = 1$ implies the target has finally been revealed.

– *Average number of successful authentication session per character in the secret words:* This metric is denoted by T. It is defined as the average number of successful authentication sessions that can be used per character in the secret words, given that the security requirement is satisfied. T reflects the user-friendliness in terms of renewal period. The longer the average usage time per character, the less frequently the user has to renew his secrets, and hence the higher the user-friendliness.

4.2 Attack Set Cardinality

In this section, we provide the final expressions in Table 1. Derivations are omitted due to space limitations. We comment that the only attack taking response as the target in PAS is random guessing as the response tables vary in each round. We also comment that the expressions of $|\mathcal{S}|$ for SAT attack are obtained right before secret renewal. This indicates the PAS scheme masks the real predicates used in each round very well. Even a SAT attack that makes perfect use of information obtained from previous observation can only reveal a large set of possible predicates, and hence the secret, right before renewal. By *right before renewal*, we mean the SAT attack has observed all successful authentication sessions the user can have, *i.e.*, the knowledge it has achieves maximum. The cardinality of $\mathcal{S}$ of secrets and predicates under SAT attack is the smallest in Table 1.

4.3 Average Usage Time per Character

In this section, we will derive the average usage time per character, T. Denote t_{max} as the maximum number of authentication sessions the same predicate indices can be used under security requirement, the expression of T can be given by $T = t_{max}/pk$. This is because the predicate indices used for one authentication session contains $p \cdot k$ terms, each of which consists of one character. In the following, we show how to obtain t_{max}.

As the SAT attack is the most effective one targeting secrets and predicates, t_{max} is determined by $t_{max} = min\{t^{sec}_{max}, t^{pre}_{max}\}$, where t^{sec}_{max} is the maximum value of t satisfying security requirement when secrets are attack target, and t^{pre}_{max} is the maximum value of t satisfying security requirement when predicates are the attack target.

Let S^r_{min} denote the desired secure requirement (in terms of attack set cardinality) against temporary impersonation by guessing responses. Let S^p_{min} denote the desired secure requirement (in terms of attack set cardinality) against permanent impersonation by revealing preset secret. Let S^t_{min} denote the desired secure requirement (in terms of attack set cardinality) against temporary impersonation by revealing predicates. S^p_{min} and S^t_{min} indicate the desired space of possible secrets and predicates under the perfect SAT attack right before the renewal.

Since t^{sec}_{max} is the maximum value of t such that security requirement S^p_{min} is satisfied under SAT attack on secret, its expression can be given by reforming the corresponding formulas in Table 1 as below,

$$S^p_{min} \leq [M\big(1-(1-\frac{1}{M})^{N^{sec}_{min}}\big)^{\frac{len}{k}}]^{pk}|H|^{p\cdot len},$$

where $N^{sec}_{min} = \frac{kp(M|H|)^{kp}}{(k!)^p} \cdot 2^{-\ell n_r t^{sec}_{max}}$.

Similarly, expression of t^{pre}_{max} can be given by reforming corresponding formulas in Table 1 as below,

$$S^t_{min} \leq \Big\{[M\big(1-(1-\frac{1}{M})^{N^{pre}_{min}}\big)^{\frac{len}{k}}]|H|\Big\}^{kp}/(k!)^p,$$

where $N^{pre}_{min} = \frac{kp(M|H|)^{kp}}{(k!)^p} \cdot 2^{-\ell n_r t^{pre}_{max}}$.

4.4 Numerical Results

In this section, we first give the cardinality of the attack set $\mathcal{S}$ in Table 1 under default parameters. We then evaluate the sensitivity of average usage time per character T under various parameters, and point out the working space of the parameters that can achieve required security. Finally we compare our PAS scheme with the scheme in [24].

We first consider the brute force attack. When the attack target is the secret, the cardinality of $\mathcal{S}$ is 2^{103} under default parameters where $M = 5 \times 5 = 25$, $H = \{A, B, \cdots, Z\}$ ($|H| = 26$), $p = 2$, $len = 10$ and $k = 1$. It is larger than that of the cryptographically strong $2^{60} \sim 2^{100}$ [14]. Note that the size against such attacks can be relaxed to be 2^{30} when CAPTCHA is incorporated [8], and PAS actually gives results only after multiple rounds. Our PAS scheme built under the above setting is clearly secure.

We now consider the random guessing attack. Since the expressions for $|\mathcal{S}|$ when secret or predicates become the attack target under such attack are the same as those under brute force attack. We only discuss the case when responses are the attack target. Under default parameters where $n_r = 5$ and $\ell = 2$, the cardinality of $\mathcal{S}$ is 2^{10}. Considering in many applications the server can block the user after certain number of continuous failed attempts (for instance, in Chase Bank's login system, the account will be locked after authentication fails three consecutive times), this set size is considered enough in many civilian applications. Higher security can be achieved via more rounds at the cost of longer authentication time.

We then consider SAT attacks. The default value for t, the number of sessions a predicate index can be used, is two. Under default parameters, the cardinality of $\mathcal{S}$ is still about 2^{50} when secret are the attack target and still about 2^{10} when predicates are the attack target, under a perfect SAT attack right before renewal.

In the following, we will evaluate the sensitivity of average usage time per character T under various parameters, and point out the working space of the parameters. In Fig.

Table 1. Expressions of Attack Set Cardinality $|\mathcal{S}|$

	Secrets	Predicates	Responses
Brute Force	$M^{pk}\|H\|^{p\cdot len}$	N/A	N/A
Random Guessing	$M^{pk}\|H\|^{p\cdot len}$	$\frac{\left(M\|H\|\right)^{kp}}{(k!)^p}$	$2^{\ell n_r}$
SAT	$[M\big(1-(1-\frac{1}{M})^N\big)^{\frac{len}{k}}]^{pk}\|H\|^{p\cdot len}$, where $N=\frac{kp(M\|H\|)^{kp}}{(k!)^p}\cdot 2^{-\ell n_r t}$	$\Big[\Big(M(1-(1-\frac{1}{M})^N)^{\frac{len}{k}}\Big)\|H\|\Big]^{kp}/(k!)^p$, where $N=\frac{kp(M\|H\|)^{kp}}{(k!)^p}\cdot 2^{-\ell n_r t}$	N/A

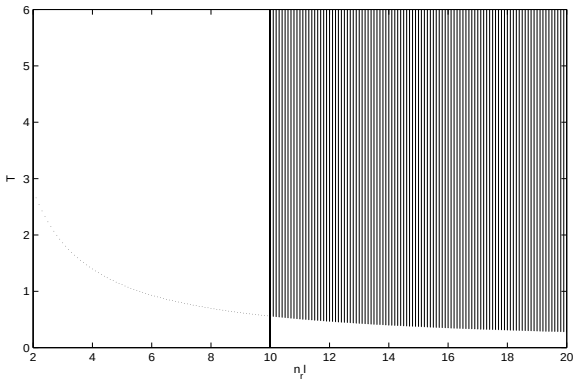

Figure 3. The shaded area shows the working space of the PAS scheme, where the parameter settings of the points on the dotted line are $M = 25, H = \{A, \cdots, Z\}$, $p = 2, k = 1, len = 10$. **The minimum secure requirements are** $S^r_{min} = 2^{10}$, $S^t_{min} = 2^{10}$ **and** $S^p_{min} = 2^{30}$.

3, although the value of $n_r\ell$ should take integers in implementation, here we include fractional values for illustration purpose. The dashed line illustrates the tradeoff between average usage time per character (T) and the number of challenge tables in each authentication session ($n_r\ell$), given that the required security level against SAT attack is satisfied. Given pre-shared secret space, the tradeoff stems from the fact that more tables contributing to the response will lead to more information leaking that can be taken advantage of by SAT attacks. Thus, smaller value of $n_r\ell$ implies each character can be used for more sessions. However, preventing information leakage is insufficient to build a strong system. Less information entropy obtained by V for an authentication session (*e.g.*, smaller value of $n_r\ell$) increases the risk under random guessing attack. Hence, the working space, which is shaded in Fig. 3, is decided by considering both. The points in the shaded area satisfy the security requirement, but may represent different parameter settings. Specifically, any point in the area above the dashed curve represents a parameter setting that is secure against SAT attack, while any point to the right of the line decided by $n_r\ell = 10$ represents a setting that is safe against random guess attack. Note that under default parameters where $n_r\ell = 10$, the number of successful sessions per character is above 0.5. Thus, with $p = 2$ and $len = 10$, the user can securely use PAS to login for at least $p \cdot len \cdot 0.5 = 10$ successful sessions before renewal under the *same* powerful passive adversary. That is, the user can securely use PAS to login for at least 10 times in exactly the same computer that has been compromised.

5 Prototype System and Usability Validation

5.1 Prototype System

We implemented a prototype system of PAS and carried out real user experiments to check usability and to explore the ways to improve it. [1]

We set up a web server with Apache Tomcat 6.0 on a PC with a 2.66 GHz Intel Pentium Core Duo CPU and 2 GB RAM. The web pages and authentication scheme were developed with JSP (JavaServer Pages) 2.1 and JDK (Java SE Development Kit) 6.0. In our prototype, the number of secrets $p = 2$, the number of terms in a predicate $k = 1$, the number of rounds in one session is adjustable from $n_r = 2$ to $n_r = 5$, the number of challenge tables in each round $\ell = 2$, the number of rows in each challenge table $m = 5$, the number of columns in each challenge table $n = 5$, character set $H = \{A, \ldots, Z\}$ and the number of characters in each cell $\tau = 13$. Figs. 1 and 2 show screenshots of our prototype. Note that the characters shown in each cell are sorted, and we color each cell at different location with a unique background color. This color will not change with tables, rounds and sessions. Thus, PAS users then are able to quickly pinpoint the cell where they look up the challenge tables.

5.2 Usability Validation

We recruited 92 participants with 8 between ages 15–18, 62 between ages 18–39, 16 between ages 40–65 and 6 between ages 65–69. Each participant was asked to read the *How to Login* page provided in our prototype before being required to finish login attempts successfully at least once for each round choice, *i.e.*, at least one successful login for $n_r = 2, 3, 4$ and 5, respectively. The results of our real user experiment validate that our PAS system has much better overall usability than existing solutions in [4, 10, 11, 23–25].

The experimental results show that PAS can be handled by all recruited users. We report the average data over all participants, since there is no salient difference in performance among age groups. For a session with two rounds, the average time for a successful login is 55.53 seconds; with three rounds, 66.03 seconds; with four rounds, 75.86 seconds; with five rounds, 84.23 seconds. More details of

[1] Link: http://drtcl4.cse.ohio-state.edu/B/ .

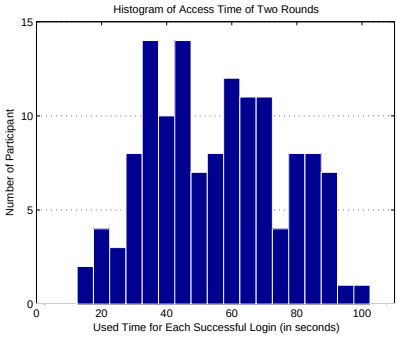

Figure 4. Successful login time for two rounds. Standard deviation: 20.34 .

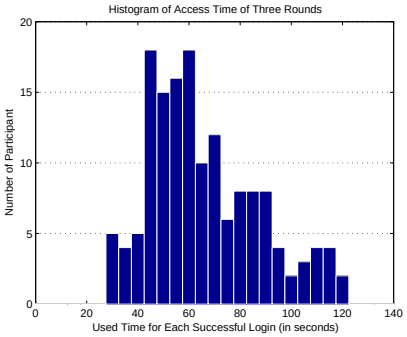

Figure 5. Successful login time for three rounds. Standard deviation: 21.76 .

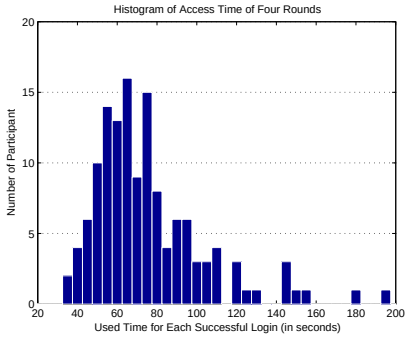

Figure 6. Successful login time for four rounds. Standard deviation: 28.3 .

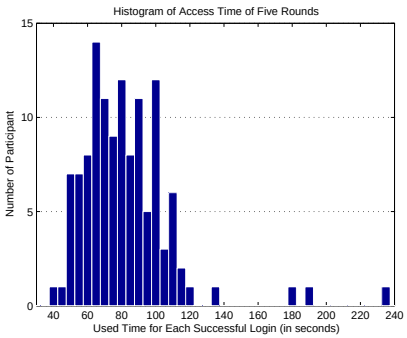

Figure 7. Successful login time for five rounds. Standard deviation: 27.0 .

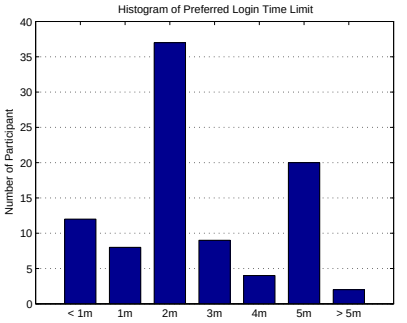

Figure 8. Preferred upper bound of login time in untrustworthy environments.

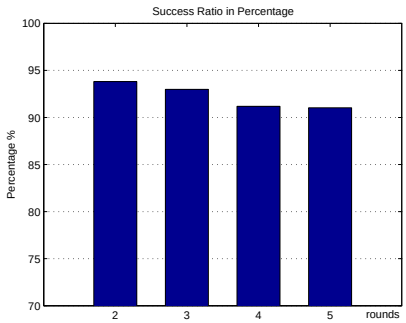

Figure 9. Login successful ratios for sessions with different rounds.

login time distribution are presented in Figs. 4–7. It is unsurprising that the time needed increases as the round number increases. One observation is that, in spite of 55.53 seconds needed to finish first two rounds, only a small amount of time around 10 seconds is needed to finish an extra round. When the number of round increases from two to three, we need extra 10.5 seconds on average; when the number of round increases from three to four, we need extra 9.83 seconds on average; when the number of round increases from four to five, we need 8.37 seconds more on average. It seems the first two rounds require more time to finish. In fact, 55.53 seconds contains the time used for a user to figure out which character should be used at the beginning of each session. Suppose each round costs around 10 seconds, this "initialization" time of the user for each session then can be expected to be around 35 seconds on average. Once the characters are figured out, they will not change in all the following rounds in a session. Since the cell locations are also fixed (with a fixed and unique color in our prototype system) in all the following rounds, the user will get used to the procedure, *i.e.*, locating the cell and telling if the character is shown, very quickly, and accomplish it faster with more practice.

We surveyed the participants about the upper bound of the login time they prefer in untrustworthy environments. The results presented in Fig. 8 show that, although traditional password based authentication needs only several seconds, it is acceptable for most people to extend login time to several minutes in untrustworthy environments. We also notice that very few people are willing to take more than 5 minutes to login. The results reflect the fact that most people are willing to trade small amount of time for security, and also validate the login time in PAS is acceptable.

The login successful ratios (number of sessions the participants successfully logged in / total number of sessions the participants tried) for authentication sessions with different rounds are shown in Fig. 9, respectively. We notice there is a slightly decrement as the number of rounds in a session increases. Overall, they are satisfactory with the least successful ratio above 90%.

PAS does not need specific training time before login process and easy to learn and use. 87 out of 92 (94.57%) participants consider PAS easy to use.

6 Conclusions

In this paper, we introduce a new concept, the *predicate*, based on which we provide the design of new authentication scheme called Predicate-based Authentication Service (PAS) against powerful passive adversaries. Using analysis and experiments on our prototype, we demonstrate that PAS can balance security and usability. As such, PAS is a candidate for a human authentication scheme that can be rapidly

adopted against powerful passive adversaries for use in untrustworthy environments.

In our research, however, we also found some limitations on PAS. First, checking a predicate once generates only 1 bit information, yes or no. This potentially increases the log in time to defend random guess attacks. One focus of our ongoing research is to obtain multiple bits output for a single and easy human operation by extending the concept of predicate or integrating it into others. Second, we found that from experiments that people are still prone to write the secret words down to find the ith character. This brings the concern on security. Furthermore, some participants consider checking tables repeatedly is boring. In our ongoing research, we are introducing images into predicate construction to further improve security and usability.

Acknowledgments

We thank anonymous reviewers for their constructive and helpful comments.

This work is supported in part by the US National Science Foundation (NSF) CAREER Award CCF-0546668, the Army Research Office (ARO) under grant AMSRD-ACC-R 50521-CI. This research does not reflect the views of the funding agencies.

References

[1] O. Goldreich, *Foundations of Cryptography: Volume 1,* Cambridge University Press, 2001.

[2] P. Golle and D. Wagner, *Cryptanalysis of a Cognitive Authentication Scheme,* in Proc. of IEEE Symposium on Security and Privacy (S&P), 2007.

[3] N. Haller, *Internet RFC 1760,* in Proc. of Symposium on Network and Distributed Systems Security (NDSS), 1994.

[4] N. J Hopper and M. Blum, *Secure Human Identification Protocols,* in Proc. of The International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), 2001.

[5] *http://world.std.com/ reinhold/diceware.html.*

[6] *http://pintday.org/advisories/misc/bwc-990420.html.*

[7] *http://www.news8.net/news/stories/1107/476003.html.*

[8] S. Li and H. Y Shum, *Secure Human Computer Identification against Peeping Attacks (SecHCI),* Cryptology ePrint Archive, Report 2005/268, 2005.

[9] G. Miller, *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information,* Psychological Review, vol 63, 1965.

[10] T. Matsumoto, *Cryptographic Human Identification,* in Proc. of the 6th International Conference on Human Computer Interaction (HCI International), 1995.

[11] T. Matsumoto, *Human-Computer Cryptography: An Attempt,* in Proc. of ACM Conference on Computer and Communications Security (CCS), 1996.

[12] F. Massacci and L. Marraro, *Towards the Formal Verification of Ciphers: Logical Cryptanalyis of DES,* in Proc. of Federated Logic Conferences (FLOC), 1999.

[13] F. Massacci and L. Marraro, *Logical Cryptanalysis as a SAT Problem,* Journal of Automated Reasoning, vol 24, 2000.

[14] A. J Menezes, etc., *Handbook of Applied Cryptography,* CRC Press Series on Discrete Mathematics and Its Applications, CRC Press, Inc., 1996.

[15] M. Mannan and P. Oorschot, *Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer,* Financial Cryptography and Data Security (FC'07), 2007.

[16] J. M McCune, A. Perrig, and M. K Reiter, *Bump in the Ether: A framework for securing sensitive user input,* USENIX Annual Technical Conference, 2006.

[17] S. Man , D. Hong, and M. Mathews, *A Shouldersurfing Resistant Graphical Password Scheme,* in Proc. of International Conference on Security and Management (ICSM), 2003.

[18] McCune et al. *Seeing-is-believing: Using camera phones for human-verifiable authentication,* IEEE Symposium on Security and Privacy (S&P), 2005.

[19] B. Parno, C. Kuo, and A. Perrig. *Phoolproof phishing prevention,* In Financial Cryptography and Data Security (FC), 2006.

[20] A. D Rubin, *Independent One-time Passwords,* Computing Systems, 9(1), 1996.

[21] SAT solver competition 2007, *http://www.cril.univ-artois.fr/SAT07/results/globalbybench.php?idev=11 &idcat=61.*

[22] E. Vogel and M. Machizawa, *Neural Acitivity Predicts Individual Differences in Visual Working Memory Capacity,* Nature, 428, April, 2004.

[23] D. Weinshall and S. Kirkpatrick, *Passwords Youll Never Forget, but Cant Recall,* in Proc. of ACM Conference on Human Factors in Computing Systems (CHI), 2004.

[24] D. Weinshall, *Cognitive Authentication Schemes Safe Against Spyware,* in Proc. of IEEE Symposium on Security and Privacy (S&P), 2006.

[25] C. H Wang, T. Hwang, and J. J Tsai, *On the Matsumoto and Imai's Human Identification Scheme,* in Advances in Cryptology - Eurocrypt, vol 921, 1995.

[26] S. Wiedenbeck, J. Waters, L. Sobrado, and J. Birget, *Design and evaluation of a shoulder surfing resistant graphical password scheme,* in Proc. of the Working Conference on Advanced Visual Interfaces, Italy, 2006.

[27] H. Zhao and X. Li, *S3PAS: A Scalable Shoulder-Surfing Resistant Textual-Graphical Password Authentication Scheme,* in Proc. of 21st international Conference on Advanced Information Networking and Applications Wprkshops, 2007.