Human Identification Through Image Evaluation Using Secret Predicates*

Hassan Jameel[1], Riaz Ahmed Shaikh[1], Heejo Lee[2], and Sungyoung Lee[1]

[1] Department of Computer Engineering, Kyung Hee University, 449-701 Suwon, South Korea
{hassan, riaz, sylee}@oslab.khu.ac.kr,
[2] Department of Computer Science and Engineering, Korea University Anam-dong, Seongbuk-gu, Seoul 136-701, South Korea
heejo@korea.ac.kr

Abstract. The task of developing protocols for humans to securely authenticate themselves to a remote server has been an interesting topic in cryptography as a replacement for the traditional, less secure, password based systems. The protocols proposed in literature are based on some underlying difficult mathematical problem, which are tuned so as to make them easily computable by humans. As a result these protocols are easily broken when desired to be efficiently executable. We present a Human Identification Protocol based on the ability of humans to efficiently process an image given a secret predicate. It is a challenge-response protocol in which a subset of images presented satisfies a secret predicate shared by the challenger and the user. We conjecture that it is hard to guess this secret predicate for adversaries, both humans and programs. It can be efficiently executed by humans with the knowledge of the secret which in turn is easily memorable and replaceable. We prove the security of the protocol separately for human adversaries and programs based on two separate assumptions and justify these assumptions with the help of an example implementation.

1 Introduction

The problem of constructing human identification protocols is an important one in the cryptographic community. That is to say, how can a human H authenticate to a remote server C, without using any computational aid? To make matters worse, the communication link between H and C is controlled by an adversary who can either passively listen to their conversation or actively interfere at will. Under such conditions, it is desirable that this adversary should not be able to masquerade as H even after observing a number of authentication sessions.

* This research was supported by the MIC (Ministry of Informations and Communications), Korea under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) incollaboration with SunMoon University. The corresponding author is Dr. Sungyoung Lee.

M. Abe (Ed.): CT-RSA 2007, LNCS 4377, pp. 67–84, 2007.

Notice that traditional password based schemes are completely insecure in this environment, in the sense that even the remote terminal which is being used by H to perform the authentication protocol is not trusted. It might contain malicious software like key-loggers, that can grab and record everything that H types, or someone might be using a hidden camera to see the alpha-numerics being typed. Ideally, H should be presented with a set of challenges, which H processes efficiently with the help of some shared secret and replies such that the responses yield little or nothing about the secret. The amount of processing required on H's part defines the feasibility of the protocol. What kind of protocol is practical for humans? We are not particularly good at numerical calculations; however we can be reasonably proud of our image processing abilities. If we can pose a challenge that involves evaluating an image based on some secret criteria then we might be able to construct a human executable protocol.

Consider the example of an IQ test based on images. The purpose of such a test is to check a person's intellectual abilities. The hardness of these questions is relative; for some these tests are harder than for others. But once a particular hint for a question is given, it would be answered promptly unlike someone who has to solve the question without any aid. If we assume that the hint to the question is a secret shared between the questioner and the human, then those who know the secret hint before hand can reply instantly whereas others without the knowledge of the hint would take considerably longer time. Furthermore, since we are involving images, the task of writing a computer program to answer these questions seems extremely hard as well, much like the automated CAPTCHA tests [6] that most humans can solve but computer programs cannot pass with a certain probability. The absence of such a "hint" makes the task of guessing the answer hard even for a human adversary. Additionally and more importantly, this "hint" can be renewed after even a small number of authentications as it will not be hard for a human to remember a natural statement.

The idea of constructing secure and efficiently executable human identification protocols using human's cognitive abilities about picture processing is not new. Matsumoto and Imai [1] were the first to propose a human identification protocol, followed by Wang *et al* [2], Matsumoto [3], Yang Li and Teng [4], Hopper and Blum [5] and Li [7]. All of these schemes can be implemented graphically. The idea is to map the secret with a set of images. The user only has to remember the images instead of a string of secret. This allows for easy secret recollection as humans can easily recall images as compared to textual strings. However, the security of the scheme relies on some underlying numerical problem like the Learning Parity with Noise (LPN) problem in [5] and the images are only used as an aid for learning the secret. Due to this fact, the complexity of these schemes is still a bit high, e.g. the HB protocol of [5] requires more than 300 seconds on average by humans. More recently, Weinshall [12] has proposed a scheme based on the SAT problem, which was subsequently broken in [13]. Notice that even in their scheme, the secret is a set of prespecified grid positions and in order to easily memorize these positions, pictures are introduced. They do not make use of the underlying structure of the images apart from using them as a memory aid. DeJa Vu[8] and Passface [9] are

purely graphical authentication schemes in which the user is asked to remember a subset of secret pictures and is then asked to authenticate by choosing his/her chosen images in a pool of pictures. A similar concept is to point and click secret locations in an image as in [10]. Apparently, in these schemes the user does not have to do any computational effort whatsoever. But this does not come without a drawback. These systems are not secure if someone is observing the actions of the user. If an adversary monitors the users' selections, it will be pretty easy to learn the secret images or locations. We argue that an image itself has a very complex structure and features, and instead of using it just as a memory aid, we can use the internal properties of images to pose challenges that can be efficiently executed by humans. Additionally, we might relax the required number of authentication sessions with a given secret, if the secret is easy to remember by a human and can be changed without too much effort on H's part. From the previous efforts, we can see that apart from the usual trade off between security, secret size and computational time, we face the problem that if we want to renew the secret after a small number of authentication sessions, it seems hard for a human to remember the new secret immediately.

In this paper, we show that it is possible to construct a challenge-response protocol, each challenge of which contains a kind of AI hard problem like CAPTCHA [6] for a computer algorithm. Moreover, the challenge is presented in such a way that a human without the knowledge of the secret can make "little sense" of the correlation between the elements of the challenge. We claim that such hard problems exist, and a protocol based on these problems will be very hard to break for adversaries, both humans and computer programs, while being efficiently executable for the legitimate user. Exact quantification of the hardness of these problems however remains an open issue.

2 An Informal Description of the Protocol

The central idea proposed in this paper is informally as follows: How can we combine the notion of CAPTCHA (creating a challenge-response that is not susceptible to bots) and secure user authentication that is not vulnerable to shoulder surfing or sniffing? To accomplish this feat, we propose the following protocol:

SETUP. User and the server agree upon a secret that is a composite of the following:

1. A simple question Q (which we will call a predicate) with only a binary answer, such as "Does the picture contain a woman?"
2. A set of distinct random numbers $a_1, a_2, \ldots, a_r$; all between 1 and n.

SERVER TO USER. A list of n pictures that are uniformly distributed with respect to the question Q above.

USER TO SERVER. n-bit binary string such that for pictures numbered $a_1, a_2, \ldots, a_r$ the corresponding bits are answers to the secret question Q and for all the other positions the bits are random.

The server accepts if the answer string is correct at the designated places. We can do the online step repeatedly to amplify security. In the full version of the protocol in Section 4, we permute the a_i's to make it harder for the adversary to extract any information from the answer string. The probability of guessing the correct permutation would be far less than that of guessing a correct random ordering of numbers. Security is based on the fact that (i) a bot or a computer program does not know the relationship between the pictures, and (ii) a human watching the proceedings would not know which bits are significant, which in turn will make it hard to guess the question being answered.

3 Definitions

We start with a set of definitions formalizing the notions of Identification Protocols and the new concepts introduced in this paper. We first define the notion of an *AI problem solver* which is modified from the definition of an *AI problem* in [6].

Definition 1. *An AI problem solver is a function $f : S \rightarrow R$, where S is a set of AI problem instances and $R \in \{0,1\}^*$ is the answer alphabet. A family of AI problem solvers is a map $F : \mathsf{Keys}(F) \times S \rightarrow R$. Here $\mathsf{Keys}(F)$ denotes the set of all AI problem solvers from S to R. Namely, if $k \in \mathsf{Keys}(F)$ then $F_k : S \rightarrow R$ is an AI problem solver.*

Notice that we specifically define a family of AI problem solvers instead of just a single one. Such a family will allow us to distribute different secrets, namely $k \in \mathsf{Keys}(F)$, to different users for authentication. The concept is similar to a function family.

We define the (δ, τ)-hardness of an AI problem solver similar to [6]:

Definition 2. *An AI problem solver f is said to be (δ, τ)-solved if there exists a program A, running in time at most τ, on an input $s \overset{R}{\leftarrow} S$, such that*

$$\Pr\left[s \overset{R}{\leftarrow} S : A(s) = f(s)\right] \geq \delta$$

f is said to be (δ, τ)-hard if no current program is a (δ, τ)solution to f, and the AI community agrees that it is hard to find such a solution.

Definition 3. *A family of AI problem solvers is said to be (δ, τ)-hard if for all keys $k \in \mathsf{Keys}(F)$, F_k is (δ_k, τ_k)-hard with $\delta_k \leq \delta$ and $\tau_k \leq \tau$.*

Definition 4. *We say that a function family, $F : \mathsf{Keys}(F) \times S \rightarrow R$ is $(\lambda(r), \tau)$-resilient against key recovery, if for all H running in time at most τ, we have:*

$$\begin{aligned}\Pr[&k \overset{R}{\leftarrow} \mathsf{Keys}(F); b_1 b_2 \ldots b_r \overset{R}{\leftarrow} \{0,1\}^r; \\ &s_1 s_2 \ldots s_r \leftarrow S | \, F_k(s_1) F_k(s_2) \ldots F_k(s_r) = b_1 b_2 \ldots b_r; \\ &k' \leftarrow H(s_1 s_2 \ldots s_r) : k = k'] < \lambda(r)\end{aligned}$$

Notice that H is not shown the value of the function F_k at each of the 'r' inputs. H only knows that the answer to each input belongs to the range R. It has to guess the correlation between the different inputs. Of course, the inputs must have an internal structure in order for the above definition to make sense. We do not elaborate this correlation between the inputs here as it will become clear when we describe the security of our protocol against human adversaries, instantiated with a suitable choice of the function family F in Section 6.

We restate the definitions of identification protocols and human executable protocols from [5] for reference:

Definition 5. *An Identification Protocol is a pair of probabilistic interactive programs (H, C) with shared auxiliary input z, such that the following conditions hold:*

- *For all auxiliary inputs z,* $\Pr\left[\langle H\left(z\right), C\left(z\right)\rangle = accept\right] > 0.9$
- *For each pair $x \neq y$,* $\Pr\left[\langle H\left(x\right), C\left(y\right)\rangle = accept\right] < 0.1$

When $\langle H, C\rangle = accept$, we say that H authenticates to C. The transcript of C contains challenges c and that of H comprises responses r to these challenges.

Definition 6. *An Identification Protocol (H, C) is (α, β, t)-human executable if at least a $(1-\alpha)$ portion of the human population can perform the calculations H unaided and without errors in at most t seconds with probability greater than $(1-\beta)$.*

Discussion on the definitions. We have separately defined a family of AI problem solvers and an $(\lambda\left(r\right), \tau)$-resilient function family. This partition is due to the fact that we want different security assumptions for a program and a human adversary. For an adversarial program, we require that the actual hardness in breaking our protocol relates to solving a function from the family of AI problem solvers. The set of keys '$\mathsf{Keys}(F)$' need not be hidden from this program or more strongly, the program might even be given the particular key being used and asked to guess the value of the function at a new input. We conjecture that such a program will still not be able to succeed but with a small probability. More details will follow in the next sections. On the other hand, we require a rather weak security assumption for human adversaries. Obviously, for a human the AI problem solvers will not pose any problems by definition. Instead, we present a $(\lambda\left(r\right), \tau)$-resilient function family to the human adversary. More specifically, we hide the set $\mathsf{Keys}(F)$ from the human adversary; randomly select a key from this set; draw a set of r inputs at random and ask the human adversary to guess the key. Since we assume the function family is $(\lambda\left(r\right), \tau)$-resilient, the probability of guessing the key is very small. We will present a function family F and argue that it satisfies both these requirements. The same function family F can both be a family of AI problem solvers and $(\lambda\left(r\right), \tau)$-resilient at the same time. Indeed, we give an example of such an F.

4 Proposed Protocol

The main theme of our protocol as described in Section 2 is to present a human with a series of pictures that satisfy a certain predicate. The user answers a pre-specified set of these pictures in an order determined by a hidden secret permutation. The assumption is that the analysis of these pictures by a computer program is extremely hard and even for a human adversary, the challenge of guessing the secret predicate when the answers are given in a random order, seems implausible. We first give a description of the protocol based on a generic building block F, and give a detailed practical example of this building block in Section 6.

PRELIMINARIES. The following notations and functions will be used in the protocol and hence forth.

$\mathsf{Perm}\,(L, l)$: Outputs the hidden permutation string $P = q_0^* p_1 q_1^* p_2 q_2^* \cdots p_l q_l^*$ of length L, obtained by first randomly selecting a permutation of the set $\{1, 2, \ldots, L\}$ and then randomly selecting $L - l$ locations in this permuted string and replacing the numbers in the corresponding locations by 0's. The non-zero numbers are $p_1, p_2, \ldots, p_l$. Each q_i^* is either null or a substring of 0's and $|q_0^*| + |q_1^*| + \cdots + |q_l^*| = L - l$.

$\mathsf{Insert}\,(P, a_1 a_2 \ldots a_l)$: Given $P = q_0^* p_1 q_1^* p_2 q_2^* \cdots p_l q_l^*$ and an l-bit binary string $a_1 a_2 \ldots a_l$, outputs a binary string $b_0^* a_1 b_1^* a_2 b_2^* \cdots a_l b_l^*$, where each b_i^* is a random binary substring whose length is equal to q_i^*.

Note. Notice that in this procedure the human user has to input random bits at the positions of q_i^*'s. This is an idealized assumption since humans may not be able to generate truly random bits. We acknowledge this as a drawback but use it nevertheless since it makes our analysis simpler.

$\mathsf{Sep}\,(P, r)$: From the given $P = q_0^* p_1 q_1^* p_2 q_2^* \cdots p_l q_l^*$ and an L-bit binary string $r = r_1 r_2 \ldots r_L$, does the following:

- $r' \leftarrow null$
- For $k = 1 \text{to} L$
  - If $P[k] \neq 0$
    - $r' \leftarrow r' || r\,[k]$
    - Output r'

Notice that, $|r'| = l$.

We are now ready to describe our protocol. After presenting the protocol based on a generic building block F and defining the security of our protocol under the assumed hardness of this building block, we will describe a suitable instantiation of this building block in the next section. It is our thesis that the proposed candidate satisfies our security requirements.

System Parameters. L, l and m; all positive integers.
Setup. H and C evaluate $\mathsf{Perm}(L, l)$ and keep the resulting hidden permutation P as a shared secret. From a family of (δ, τ)-hard AI problem solvers $F : \mathsf{Keys}(F) \times S \to \{0, 1\}$, H and C randomly select a secret key $k \in \mathsf{Keys}(F)$. C also sets $S^- = \{\}$.
Protocol.

- Set $j = 0$
- While $j \neq m$ or $j \neq \perp$:
  - C randomly chooses a binary sequence $b_1 b_2 \cdots b_L$ from $\{0, 1\}^L$.
  - for $i = 1$ to L:
    - If $b_i = 1$, C selects a random $s_i \in S - S^-$ such that $F_k(s_i) = 1$ and updates $S^- \leftarrow S^- \cup \{s_i\}$.
    - Else C selects a random $s_i \in S - S^-$ such that $F_k(s_i) = 0$ and updates $S^- \leftarrow S^- \cup \{s_i\}$.
    - Csets $s = s_1 s_2 \cdots s_L$ and sends it to H.
  - For $i = 1$ to l:
    - H computes $F(s_{p_i})$. If it is 1, it assigns $a_i = 1$ otherwise assigns $a_i = 0$.
  - H sends $r' = \mathsf{Insert}(P, a_1 a_2 \ldots a_l)$ to C.
  - If $\mathsf{Sep}(P, r') = F(s_{p_1}) \parallel F(s_{p_2}) \parallel \ldots \parallel F(s_{p_l})$, C increments j otherwise C sets $j = \perp$.
- If $j = m$, C accepts H.

It is easy to see that the probability of someone impersonating a legitimate user by randomly submitting answers is 2^{-lm}. We have defined the set S^-, so that once an input from a set S has been used, it should no longer be used again for that particular user. In practice we can define two sets: S_1 and S_0 denoting the sets whose elements evaluate to $F_k(.) = 1$ and 0 respectively. Each time an input is used from this set, it is taken out of this set and never used for the same user. The reason and practicality of this requirement will become clear when we show a reasonable instantiation.

5 Security Analysis

5.1 Security Against Passive Adversarial Programs

An Identification Protocol (H, C) is (p, q) secure against passive adversaries if for all computationally bounded adversaries $\mathcal{A}$,

$$\Pr\left[\langle \mathcal{A}\left(T^q\left(H(z), C(z)\right)\right), C(z)\rangle = accept\right] \leq p$$

Here $T^q(.,.)$ represents the transcript of q communication sessions between H and C. In the appendix, we assume that if $\mathcal{B}$ is a program, then even after observing a certain amount of values of $F_k(.)$, it cannot solve this function with probability better than δ. With this assumption, we prove the following for our protocol:

Claim 1. If F is a (δ, τ)-hard family of AI problem solvers, then for all adversarial programs $\mathcal{A}$:

$$\Pr\left[\left(\mathcal{A}\left(T^q\left(H\left(z\right), C\left(z\right)\right)\right), C\left(z\right)\right) = accept\right] \leq 2\delta - 1$$

Proof. See Appendix A.1.

5.2 Security Against Passive Human Adversaries

For a human adversary, we assume that F is a family of $(\lambda\left(r\right), \tau)$-resilient functions. Hence it is not possible to extract the key k of a particular function F_k of this family, when less than r of the instances are shown to a human adversary. We can construct an experiment in which this adversary is also given a random binary sequence of r bits. Obviously, this adversary has no advantage in detecting the key k, if this sequence is truly independent of the choice of answers. However if we give this adversary the true sequence of answers, then it might have considerable advantage in detecting the secret key. All we need to show is that our protocol does not reveal the true answers with all but a small probability. Ideally, the answer string should be a pseudorandom binary string; however we do not know how a human would be able to generate a cryptographically secure pseudorandom string without any computational aid. The best we can do is to introduce some randomness in the answer strings based on the secret permutation and the random bits. We digress here to explain the need for secret permutation and the random bits. Consider an adversary that randomly picks up a permutation and the corresponding locations and tries to detect the key. For a suitable choice of $L = 10$ and $l = 5$, the probability that the adversary's guess was correct is $\frac{5!}{10!}/\binom{10}{5} \approx 2^{-22}$. If instead, we use a simpler procedure of randomly selecting 5 locations (as in Section 2), then the adversary's probability of guessing the hidden locations is just $1/\binom{10}{5} \approx 2^{-8}$. To achieve the same level of probability, we would have to choose $L = 24$ and $l = 12$, which means that the user has to remember more locations. This motivates the use of our procedure.

We show in Appendix A.2, that in a given round, the probability of the event that the answer string returned by H is equal to the actual answer string (without permutation and random bits) is less than $\frac{1}{2^r}\left(1 + \frac{l}{L^2}\right)^r$. Since our adversary has no advantage in solving our protocol without guessing the random permutation, it follows that the advantage of this adversary in breaking our protocol is bounded by $\lambda\left(r\right)$ and the above probability, as shown in the appendix.

5.3 Security Against Active Adversaries

We take the definition of an active adversary from [5]. We do not distinguish between a human adversary and an adversarial program in this case. *An identification protocol (H, C) is (p, q, r)-detecting against active adversaries if for all computationally bounded adversaries $\mathcal{A}$,*

$$-\Pr\left[\left\langle H\left(z\right), \mathcal{A}\left(T^r\left(H\left(z\right), C\left(z\right)\right)\right)\right\rangle \neq \perp\right] < q$$
$$-\Pr\left[\left\langle \mathcal{A}\left(T^r\left(H\left(z\right), C\left(z\right)\right)\right), C\left(z\right)\right\rangle = accept\right] < p$$

Against active adversaries we have a natural defense thanks to the cognitive abilities of humans. If a human can detect that a particular instance has been replayed twice or more with a high probability, then he may reject C, hence terminating the protocol session. Hence our protocol has an innate defense against replay attacks. More specifically, let $1-q$ be the probability that H can detect an instance s' being replayed. Obviously, this probability should be a function of time and the specific iteration at which the instance is being replayed. Let S_i^- denote the variable defining the set S^- after the ith query to C by H. Let A_i be the event that an s' was presented to H at the ith query, such that $s' \in S_j$ for some $j < i$. Then,

$$\Pr\left[H \text{ detects } A_i\right] \geq \Pr\left[H \text{ detects } A_r | s' \in S_1^- \wedge s' \notin S_r^-, 1 < j < r\right] \geq 1 - q$$

All other events will be predicted with greater probability. Thus with high probability the human will detect a replay challenge attack.

Notice however, that our protocol is not secure against another kind of attack: the automated adversary intercepts the sequence from the server to the user, and replaces one instance in that sequence by some other instance (taken from some other source). If the user's reply is rejected, the adversary now has two instances for which it knows that the answers are different. After collecting a few such pairs, all these pairs are displayed to a human adversary, which now has the instances alongwith their answers. This attack, however would result in the termination of the session, if successful. The protocol could be repaired to handle this kind of active attack. Namely, we can allow the user to submit wrong answers about half of the time. This would result in an increase in the number of rounds and would take more time. We acknowledge it as a weakness in our protocol and leave a possible efficient fix as a future work. There may be other active attacks from a combined human-computer adversary and a thorough analysis is certainly required.

6 A Suitable Instantiation of F

We describe a procedure that seems a plausible candidate for an AI problem solver as well as being $(\lambda(r), \tau)$-resilient at the same time. Our motivation is the saying that a picture is worth a thousand words. A picture might satisfy a variety of predicates. Consider as an example, the picture in Figure 1. How many different predicates does this picture satisfy? To list a few:

- Does this picture present a cartoon?
- Is there a "nose" in this picture?
- Is there a woman in this picture?

And so on. In short, we can select a predicate, find a set of pictures that satisfy this predicate and a set that does not. We present these pictures to a human user in place of F in our protocol and we are sure that it will satisfy our goals. Let Pic denote a collection of pictures and let *pic* be a member of it. Let Pred denote the set of all predicates. We define the family of functions:

$$Q : \mathsf{Pred} \times \mathsf{Pic} \rightarrow \{0, 1\}$$

Fig. 1. A picture might represent a large number of concepts and contexts

Conjecture 1. Q defines a family of AI problem solvers.

The CAPTCHA project has a particular CAPTCHA named ESP-PIX [11], that presents a set of pictures related to each other through some feature. The pictures are rather obviously related to each other when viewed by a human; however it is claimed that for a computer program it is extremely hard to find a common link between the pictures. We present a similar but harder problem. In our protocol, the pictures satisfying the predicate are intermingled with those that do not, and we ask a computer program to tell whether the pictures are interlinked or not.

Conjecture 2. Q defines an $(\lambda(r), \tau)$-resilient family of functions.

This claim seems hard to justify. Our claim is based on the belief that the following problem would be hard for a human. Namely, a human is given a series of pictures as in Figure 2; is told that some of these pictures satisfy a given predicate and some don't, and is asked to guess the hidden predicate. Of course without the knowledge of the answers to the predicate for each of these pictures, this seems to be a hard problem. With the knowledge of the answers, we cannot

Fig. 2. A set of pictures, some or all of which satisfy a hidden predicate

say with a certain amount of confidence that a person might not be able to guess the hidden predicate. We constructed our protocol in a way so as to hide the answers, such that the adversary might not be able to gain advantage by guessing the hidden predicate. An important question is: What kind of predicate to choose? A predicate involving color differences like "Does the picture contain

the color red?" should most certainly not be used. Color difference is very easily caught by the human eye. The predicate used in Figure 2 is "Does the object in the image begin contain the letter "P" in its name?" Predicates like this, which do not catch the human eye, are the likely candidates.

6.1 A User Friendly Implementation

For human users the parameters $L = 10, l = 5$ and $m = 4$ can be chosen. Note that a random guess attack can only succeed with a probability 2^{-20} in this case, which is more than the security of a 4-digit pin number. The user is given a hidden permutation string say: 0098030502. When the user inputs his ID, he is brought to a page containing 10 pictures in a 2X5 grid at the bottom of which is a text box. The user answers by randomly picking '0' or '1' in place of the '0's' in the permutation string and answering the pictures in the specified order corresponding to the digits other than the '0's'. So, for example, a possible answer would be 10<u>1</u>10<u>0</u>1<u>1</u>01, where the underlined digits denote the actual answers and the rest are random bits. The user would input the string 1011001101 and will go to the next series of 10 pictures if this answer is correct at the specified positions. The procedure continues until 4 steps and the user is accepted once all the 4 steps result in a success. The choice of using 10 pictures in a challenge seems appropriate, as they can easily be displayed on the screen as shown in Figure 3. A single picture would take around 5 seconds for a human

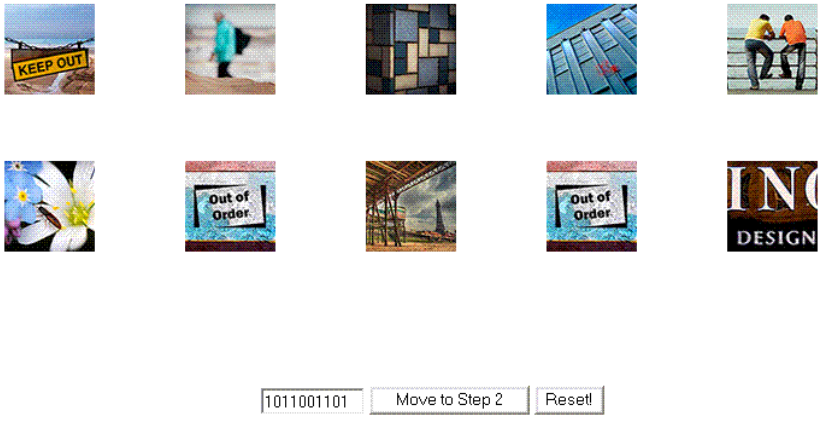

Fig. 3. An authentication step in our proposed protocol

to verify whether it satisfies the predicate or not. This would take around 100 seconds to execute the protocol. This amount of time seems reasonable if we use the protocol only under certain circumstances such as when the user is trying to log on through an insecure public terminal.

7 Limitations and Discussion

One might ask the question that how long the protocol should be run to keep a desired security level. It is evident that based on our assumption, the protocol can be safely executed for a number of times if we only consider adversarial programs. How about human adversaries? We know an inherent weakness in us humans; the more the work load, the less efficient we are. So if a human adversary is given a collection of, say 2000, pictures and is asked to find the hidden predicate, then he might not be able to examine all these pictures. In order to make the task harder for a human adversary, we can have two or more predicates connected through a truth clause. The user then checks whether the picture satisfies the clause and answers accordingly. This will result in an increase in execution time, but guessing the secret predicates will be much harder.

Another question is regarding the possible usage of our protocol. We insist that our protocol should be used only in situations when a user is away from the luxury and security of his personal computer or office environment. The user might want to use our system when using a public terminal to log in for emails while on a business trip. However, when he is back in his office or home, he can use the normal password based system to log in to his computer. So, we can safely use our system for a small number of authentications before the secret predicate can be refreshed and a new hidden permutation can be used. The fact that the secret predicate plus the hidden permutation is not a load on a human's memory (the hidden permutation being the size of a normal telephone number) makes this switch very practical and reusable. Consequently, we can use this system, for say 20 or 30 authentications before renewing the secret.

A modified version of the protocol, secure against general active adversaries is also desirable; without making it infeasible or increasing the number of rounds. The most important limitation is the selection of the predicates and selecting appropriate pictures that satisfy these predicates. This can be done by a dedicated group from the service providers. We do not know however, if this task can be performed by a computer or not. Automatically generated predicates and pictures might prove helpful and will increase the practicality of our scheme. Another important direction is to find whether there exist other functions in place of the predicate-image one. A function family whose soundness can be theoretically proved instead of being conjectured would certainly be a better candidate than the one presented in this paper.

8 Conclusion

The problem of making secure human identification protocols in which a human authenticates to a remote server has resulted in many efficient authentication protocols over the years. These protocols try to make things easy for humans by presenting them a challenge based on some mathematical problem which is easy to compute but difficult for an adversary to crack. However, the efficiency of these protocols lie in the user friendly representation. Instead of using computational problems, we can look for problems in domains where humans are better

than computers, like image evaluation. However, to construct an identification protocol, we need some problem that is not only hard for computer programs but for human adversaries too. We have shown that it is possible to construct a protocol based on human's excellent image processing abilities such that defeating the protocol is hard even for the human adversaries. The proposed problem based on evaluating an image through a secret predicate seems to be hard to crack even for human adversaries who do not have the knowledge of the secret. This allows us to deal with the security of the protocol separately for the adversarial programs and for human adversaries. It will be interesting to investigate further in search for other possible problems that satisfy this nice feature. The obvious open question and limitation is to mathematically quantify the hardness of the problem discussed in this paper. This, however, remains an open problem.

Acknowledgements

We are grateful to Stuart Haber for his comments and the anonymous reviewers for their suggestions.

References

1. Matsumoto, T., Imai, H.: Human Identification through Insecure Channel. Advances in Cryptology - EUROCRYPT 91, Lecture Notes in Computer Science, Springer-Verlag. **547** (1991) 409–421
2. Wang, C.H., Hwang, T., Tsai, J.J.: On the Matsumoto and Imai's Human Identification Scheme. Advances in Cryptology - EUROCRYPT 95, Lecture Notes in Computer Science, Springer-Verlag. **921** (1995) 382–392
3. Matsumoto, T.: Human-computer cryptography: An attempt. 3rd ACM Conference on Computer and Communications Security, ACM Press. (1996) 68–75
4. Xiang-Yang Li, Shang-Hua Teng: Practical Human-Machine Identification over Insecure Channels. Journal of Combinatorial Optimization. **3** (1999) 347–361
5. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. Advances in Cryptology - Asiacrypt 2001, Lecture Notes in Computer Science, Springer-Verlag. **2248** (2001) 52–66
6. Luis von Ahn, Manuel Blum, Nicholas Hopper, John Langford: CAPTCHA: Using Hard AI Problems for Security. Advances in Cryptology – Eurocrypt 2003, Lecture Notes in Computer Science, Springer-Verlag. (2003) 294–311
7. Shujun Li, Heung-Yeung Shum: Secure Human-computer identification against Peeping Attacks (SecHCI): A Survey. Unpublished report, available at Elsevier's Computer Science Preprint Server. (2002)
8. Rachna Dhamija, Adrian Perrig: Deja Vu: A user study using images for authentication. Proc. of the 9th USENIX Security Symposium. (2000) 45–58
9. ID Arts Inc: Passfaces - the Art of Identification. Visit http://www.idarts.com
10. Vince Sorensen: PassPic - Visual Password Management. Visit http://www.authord.com/
11. The CAPTCHA project: ESP-PIX. Visit http://www.captcha.net/
12. Daphna Weinshall: Cognitive Authentication Schemes Safe Against Spyware (Short Paper). 2006 IEEE Symposium on Security and Privacy. (2006) 295–300
13. Philippe Golle and David Wagner: Cryptanalysis of a Cognitive Authentication Scheme. Cryptology ePrint Archive, Report 2006/258. http://eprint.iacr.org/.

A Security Analysis

A.1 Security Against Passive Adversarial Programs

We first define the following oracles that represent the different functionalities in our protocol.

The oracle F_k. For any $k \in \mathsf{Keys}(F)$, this oracle takes as input an $s \in S$ and outputs $F_k(s) \in \{0,1\}$.

We assume a global set S^- initially empty, available to the following oracles.

The oracle C. This oracle takes as input the following queries:

- Init : This query initializes a new session and terminates any previous session. C randomly outputs the sequence of instances $s = s_1 s_2 \dots s_L$ from $S - S^-$ and updates $S^- \leftarrow S^- \cup s$, $j \leftarrow 1$.
- $C(a_1 a_2 \dots a_L)$: If $j = \perp$ outputs *reject*. Else if $j = m$ outputs *accept* or *reject* and updates $j \leftarrow \perp$. Else if $j < m$, yields one of two possible outputs:
  - (s) ; Randomly outputs the sequence $s = s_1 s_2 \dots s_L$ from $S - S^-$ and updates $S^- \leftarrow S^- \cup s$, $j \leftarrow j + 1$.
  - $(reject)$; Outputs $(reject)$ and sets $j \leftarrow \perp$.

The oracle H. Takes as input the query $H(s = s_1 s_2 \dots s_L)$. If $S^- \cap s = \varphi$, outputs $a_1 a_2 \dots a_l$. Else outputs $\perp$.

Now suppose an AI problem solver outputs a sequence of bits $r_1 r_2 \dots r_t$ on the inputs $s_1 s_2 \dots s_t$. An adversarial program $\mathcal{A}$ wants to guess $F_k(s_{t+1})$ on being given the challenge s_{t+1}. The following experiment describes this functionality:

Experiment $\mathbf{Exp}^{aps}_{F,\mathcal{A}}$
$k \xleftarrow{R} \mathsf{Keys}(F)$
$s \leftarrow \mathcal{A}^{F_k}$
$b \leftarrow \mathcal{A}^{F_k}(s)$
If $b = F_k(s)$ return 1 else return 0

The *aps-advantage* of $\mathcal{A}$ is defined as:

$$\mathbf{Adv}^{aps}_{F,\mathcal{A}} = \Pr\left[\mathbf{Exp}^{aps}_{F,\mathcal{A}} = 1\right]$$

For any t, q we define the *aps-advantage* of F as:

$$\mathbf{Adv}^{aps}_{F,\mathcal{A}}(t,q) = \max_{\mathcal{A}}\left\{\mathbf{Adv}^{aps}_{F,\mathcal{A}}\right\}$$

with the maximum being over all adversarial programs $\mathcal{A}$ having time-complexity t and making at most q oracle queries to the oracle F_k.

Conjecture. If F is a family of (δ, τ)-hard AI problem solvers, then for any program $\mathcal{A}$:

$$\mathbf{Adv}_{F,\mathcal{A}}^{aps}(\tau, |S| - 1) < \delta.$$

where $\delta > 1/2$.

Our belief on this conjecture is based on the definition of a CAPTCHA[6]. Even if we provide answers to the queries of an adversarial program, it will be hard for it to analyze all the pictures and categorize them into one category. The function family Q described in Section 6 shows just this.

Now let $\mathcal{B}$ be a passive adversarial program, such that:

$$\Pr\left[\left(\mathcal{B}\left(T^{q}\left(H\left(z\right), C\left(z\right)\right)\right), C\left(z\right)\right) = accept\right] > \beta.$$

This adversary can be described by the following experiment:

Experiment $\mathbf{Exp}_{F,\mathcal{B}}^{aut}$
  Initialize oracles C and H
  While state = "test"
    done $\leftarrow \mathcal{B}^{C,H}$
  $\mathcal{B}$ queries C until C outputs *accept* or *reject*.
  If C outputs *accept*
    Output '1'
  Else
    Output '0'

We define the advantage of $\mathcal{B}$ as:

$$\mathbf{Adv}_{F,\mathcal{B}}^{aut} = \Pr\left[\mathbf{Exp}_{F,\mathcal{B}}^{aut} = 1\right]$$

And *aut-advantage* of our protocol as:

$$\mathbf{Adv}_{F,\mathcal{B}}^{aut}(t, q_C, q_H) = \max_{\mathcal{B}}\left\{\mathbf{Adv}_{F,\mathcal{B}}^{aut}\right\}$$

Now, we relate the two adversaries with the help of the following claim:

Claim. We have:

$$2\mathbf{Adv}_{F,\mathcal{A}}^{aps}(t_p + t_s q_C + t_a q_H + \tau, (q_C + q_H)\, l) - 1 = \mathbf{Adv}_{F,\mathcal{B}}^{aut}(\tau, q_C, q_H)$$

Proof. We construct an adversary $\mathcal{A}_{\mathcal{B}}$ which is given an oracle F_k and runs adversary $\mathcal{B}$ as a subroutine. This adversary uses the advantage of $\mathcal{B}$ in defeating our protocol to predict the image of F_k at s.

The adversary is described as follows:

Adversary $\mathcal{A}_{\mathcal{B}}^{F_k}$
  Randomly select $P = \mathsf{Perm}(L, l)$. Run Adversary $\mathcal{B}$, replying to its oracle queries as follows:
  When $\mathcal{B}$ makes an oracle query init:

Set $j \leftarrow 1$ and randomly select $s = s_1 s_2 \cdots s_L$, where $s_i \in S - S^-$ and update $S^- \leftarrow S^- \cup \{s_i\}$. Update the sequence $p \leftarrow s_{p_1} s_{p_2} \ldots s_{p_l}$.

When $\mathcal{B}$ makes an oracle query $C\,(r' = a_1 a_2 \ldots a_L)$, do:

If $j = \perp$ output *reject*.

Else if $j = m$ and $\mathsf{Sep}\,(P, r') = F\,(s_{p_1})\,F\,(s_{p_2}) \ldots F\,(s_{p_l})$ output *accept* and update $j \leftarrow \perp$; else *reject* and update $j \leftarrow \perp$.

Else if $j < m$, yields one of two possible outputs:

(s); If $\mathsf{Sep}\,(P, r') = F\,(s_{p_1})\,F\,(s_{p_2}) \ldots F\,(s_{p_l})$, randomly output the sequence $s = s_1 s_2 \ldots s_L$ from $S - S^-$ and update $S^- \leftarrow S^- \cup s$, $j \leftarrow j + 1$ and $p \leftarrow s_{p_1} s_{p_2} \ldots s_{p_l}$.

("*reject*"); If $\mathsf{Sep}\,(\mathsf{Perm}, r') \neq F\,(s_{p_1})\,F\,(s_{p_2}) \ldots F\,(s_{p_l})$ output "*reject*" and set $j \leftarrow \perp$.

When $\mathcal{B}$ makes an $H\,(s = s_1 s_2 \ldots s_L)$ query do:

For $i = 1$ to l, $a_i \leftarrow F_k\,(s_{p_i})$

Return $r' = \mathsf{Insert}\,(P, a_1 a_2 \ldots a_l)$ to $\mathcal{B}$ as the answer.

Until $\mathcal{B}$ outputs the state 'done'.

Set $j \leftarrow 0$.

On $\mathcal{B}$'s init query, randomly select $s = s_1 s_2 \cdots s_L$, where $s_i \in S - S^-$ and update $S^- \leftarrow S^- \cup \{s_i\}$, $j \leftarrow 1$. Update the sequence $p \leftarrow s_{p_1} s_{p_2} \ldots s_{p_l}$.

For 2 to m do:

When $\mathcal{B}$ outputs $r' = a_1 a_2 \ldots a_L$ set $q \leftarrow q || \mathsf{Sep}\,(P, r')$. Randomly select $s = s_1 s_2 \cdots s_L$, where $s_i \in S - S^-$ and update $S^- \leftarrow S^- \cup \{s_i\}$, $j \leftarrow j + 1$. Update the sequence $p \leftarrow p || s_{p_1} s_{p_2} \ldots s_{p_l}$.

When $\mathcal{B}$ outputs $a_1 a_2 \ldots a_L$, set $q \leftarrow q || \mathsf{Sep}\,(P, r')$ and output *accept*.

Randomly select an s from $p = s_1 s_2 \ldots s_{lm}$ and set it as the output.

Output the corresponding bit in $q = a_1 a_2 \ldots a_{lm}$ as the response and halt.

Now, we can see that:

$$\begin{aligned}
\mathbf{Adv}^{aps}_{F,\mathcal{A}_\mathcal{B}} &= \Pr_{r \in F_k}\left[s \leftarrow \mathcal{A}_\mathcal{B}^{F_k}; b \leftarrow \mathcal{A}_\mathcal{B}^{F_k}\,(s)\,; F_k\,(s) = b\right] \\
&= \Pr\left[F_k\,(s) = b | \mathcal{B}\,\text{succeeds}\right] \Pr\left[\,\mathcal{B}\,\text{succeeds}\right] \\
&\quad + \Pr\left[F_k\,(s) \neq b |\,\mathcal{B}\,\text{fails}\right] \Pr\left[\mathcal{B}\,\text{fails}\right] \\
&= \frac{\mathbf{Adv}^{aut}_{F,\mathcal{B}}}{2} + \frac{1}{2}
\end{aligned}$$

Let t_p, t_s and t_a denote the running times of the procedures $\mathsf{Perm}\,(.,.)$, $\mathsf{Sep}\,(.,.)$ and $\mathsf{Insert}\,(.,.)$ respectively. $\mathcal{A}_\mathcal{B}$ has to perform $\mathsf{Perm}\,(.,.)$ once, $\mathsf{Sep}\,(.,.)$ a maximum of $t_s q_C$ times and $\mathsf{Insert}\,(.,.)$ a maximum of $t_a q_H$ times. Further notice that apart from these calculations, the adversary $\mathcal{A}_\mathcal{B}$ does some rather trivial calculations more than the adversary $\mathcal{B}$. Thus if the running time of $\mathcal{B}$ is bounded by τ, then that of $\mathcal{A}_\mathcal{B}$ is bounded above by $t_p + t_s q_C + t_a q_H + \tau$. Hence, by maximizing, we reach to the conclusion:

$$2\mathbf{Adv}^{aps}_{F,\mathcal{A}}\,(t_p + t_s q_C + t_a q_H + \tau, (q_C + q_H)\,l) - 1 = \mathbf{Adv}^{aut}_{F,\mathcal{B}}\,(\tau, q_C, q_H)$$

□

Theorem. If F is a (δ, τ)-hard family of AI problem solvers, then for all passive adversarial programs $\mathcal{B}$ running in time less than $\tau - \Delta t$:

$$Pr\left[\left(\mathcal{B}\left(T^{q}\left(H\left(z\right), C\left(z\right)\right)\right), C\left(z\right)\right) = accept\right] < 2\delta - 1$$

where, $q \approx \frac{|S|-1}{Lm}$ and $\Delta t \approx \left(1 + \frac{|S|-1}{l}\right) t$, with $t \ll \tau$.

A.2 Security Against Passive Human Adversaries

Assume that we have a human adversary $\mathcal{H}$. We consider the following experiment:

Experiment $\mathbf{Exp}^{hg}_{F,\mathcal{H}}$
$k \overset{R}{\leftarrow} \mathsf{Keys}(F)$;
$b_1 b_2 \ldots b_k \leftarrow \{0,1\}^k$;
$s_1 s_2 \ldots s_r \leftarrow S - S^{-}$, such that $F\left(s_1\right) F\left(s_2\right) \ldots F\left(s_r\right) = b_1 b_2 \ldots b_k$;
$k' \leftarrow \mathcal{H}\left(s_1 s_2 \ldots s_r\right)$;
If $k = k'$ return 1 else return 0.

The *hg-advantage* of $\mathcal{H}$ is defined as:

$$\mathbf{Adv}^{hg}_{F,\mathcal{H}} = \Pr_{r \in F_k}\left[\mathbf{Exp}^{hg}_{F,\mathcal{A}} = 1\right]$$

For any t we define the *hg-advantage* of F as:

$$\mathbf{Adv}^{hg}_{F,\mathcal{H}}\left(t\right) = \max_{\mathcal{H}}\left\{\mathbf{Adv}^{hg}_{F,\mathcal{H}}\right\}$$

Since F is an $(\lambda\left(r\right), \tau)$-resilient function family, we have

$$\mathbf{Adv}^{hg}_{F,\mathcal{H}}\left(\tau\right) = \lambda\left(r\right).$$

This tells us that if a human adversary is given a series of instances, and is not shown which one of them output 1 or which one of them output zero, then he has a probability of $\lambda\left(r\right)$ in successfully guessing the key. All we to show is that for a human observer, the following two situations are hard to distinguish but with a small probability: (a) Concatenated outputs of the two oracles in our protocol for a total of r/Lm authentication sessions. (b) A random set of r instances with a truly random answer bit string of the same length. In particular, consider a passive adversary $\mathcal{H}$, who listens r/Lm sessions of our protocol and gets the r instances $s_1 s_2 \ldots s_r$ together with their answers $a_1 a_2 \ldots a_r$. Let A_i denote the event that $F\left(s_i\right) = a_i$ in our protocol. We first prove the following:

Theorem. $\frac{1}{2^r} < \Pr\left[A_1 \wedge A_2 \wedge \ldots \wedge A_r\right] < \frac{1}{2^r}\left(1 + \frac{l}{L^2}\right)^r$, where $r = jL$ for some positive integer j.

Proof. Consider the event, $A_1 : F(s_1) = a_1$. Let σ denote the identity permutation and P denote the hidden permutation of our protocol, then,

$$\begin{aligned}\Pr[F(s_1) = a_1] &= \Pr[F(s_1) = a_1 | \sigma(1) = P(1)] \Pr[\sigma(1) = P(1)] + \\ &\quad + \Pr[F(s_1) = a_1 | \sigma(1) \neq P(1)] \Pr[\sigma(1) \neq P(1)] \\ &= 1 \cdot \frac{1}{L}\left(\frac{l}{L}\right) + \frac{1}{2}\left(1 - \frac{l}{L^2}\right) = \frac{1}{2}\left(1 + \frac{l}{L^2}\right)\end{aligned}$$

Similarly, we can find out that:

$$\Pr[F(s_2) = a_2 | F(s_1) = a_1] < \frac{1}{2}\left(1 + \frac{l}{L^2}\right)$$

And in general, for all $t \leq r$,

$$\frac{1}{2} < \Pr[F(s_t) = a_t | F(s_1) = a_1 \wedge \ldots \wedge F(s_{t-1}) = a_{t-1}] < \frac{1}{2}\left(1 + \frac{l}{L^2}\right)$$

Hence, $\frac{1}{2^r} < \Pr[A_1 \wedge A_2 \wedge \ldots \wedge A_r] < \frac{1}{2^r}\left(1 + \frac{l}{L^2}\right)^r$. □

We define the advantage of a passive human adversary attempting to defeat our protocol as $\mathbf{Adv}^{hg}_{P,\mathcal{H}}(\tau)$ and assume that, $\mathbf{Adv}^{hg}_{P,\mathcal{H}}(\tau) = \mathbf{Adv}^{hg}_{F,\mathcal{H}}(\tau) + \alpha(r)$; where, $\alpha(r) = M$,$(M < 1 - \lambda(r))$ when $\Pr[A_1 \wedge A_2 \wedge \ldots \wedge A_r] = 1$, and $\alpha(r) = 0$ when, $\Pr[A_1 \wedge A_2 \wedge \ldots \wedge A_r] = 1/2^r$. Assuming $\alpha(r)$, to be a linear function, it is straight forward to show that:

$$\mathbf{Adv}^{hg}_{P,\mathcal{H}}(\tau) < \lambda(r) + \frac{M}{2^r - 1}\left(\left(1 + \frac{l}{L^2}\right)^r - 1\right).$$