On Securing Communication From Profilers

Sandra Díaz-Santiago, Debrup Chakraborty

Department of Computer Science, CINVESTAV IPN
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco. México, D.F., 07360
sdiaz@computacion.cs.cinvestav.mx, debrup@cs.cinvestav.mx

Abstract. A profiling adversary is an adversary which aims to classify messages into pre-defined profiles and thus gain useful information regarding the sender or receiver of such messages. Usual chosen-plaintext secure encryption schemes are capable of securing information from profilers, but these schemes provide more security than required for this purpose. In this paper we study the requirements for an encryption algorithm to be secure only against profilers and finally give a precise notion of security for such schemes. We also present a full protocol for secure (against profiling adversaries) communication, which neither requires a key exchange nor a public key infrastructure. Our protocol guarantees security against non-human profilers and is constructed using CAPTCHAs and secret sharing schemes.

1 Introduction

Informally a spam email is an email which is not of interest to the receiver. Everyday almost every one of us finds hundreds of such spam emails waiting in our in-boxes. A spammer (who sends spam emails) generally has a business motive and most spam emails try to advertise a product, a web-page or a service. If the spam emails can be sent in a directed manner, i.e., if a spammer can send a specific advertisement to a user who would be interested in it, then the motive of the spammer would be successful to a large extent. Thus, one of the important objectives of a spammer would be to know the preferences or interests of the users to whom it is sending the un-solicited messages.

In today's connected world we do a lot of communication through emails and it is not un-realistic to assume that a collection of email messages which originate from a specific user U carries information about the preferences and interests of U. Based on this assumption a spammer can collect email information originating from different users and based on these emails try to make a profile of each user (based on their preferences or interests), and later use this profile for directed spamming.

Here we assume that given a message space an adversary aims to map each message in the message space into certain classes of its interest. Using this classification of messages the adversary can try to conclude which user is associated with which class and this is expected to reveal information regarding the profile of a given user. Thus, in the scenario of our interest we consider an adversary that classifies messages into pre-defined classes. Such an adversary would be further called as a profiler.

Other than directed spamming, there may be other motives for user profiling. Currently there has been a paradigm shift in the way products are advertised in the internet. In one of the popular new paradigm of *online behavioral advertising* (OBA) [14], internet advertising companies display advertisements specific to user preferences. This requires profiling the users. To support this big business of internet advertising, innovative techniques for user profiling have also developed. It is known that some internet service providers perform a procedure called *deep packet inspection* on all traffic to detect malware etc., but this technique has been used to generate user profiles from the information contents of the packets received or sent by an user, and this information is later sold to advertising companies [14]. This currently has led to many policy related debates, and it has been asked whether such practices should be legally allowed [11].

In the context of emails, a solution to the problem of profiling attacks would be encrypting the communications so that the contents of the emails are not available to the profiler. Or to make the communications anonymous so that given a message it would not be possible for a profiler to trace the origin of the message. In this paper we ask the following question: What would be the exact security requirements for an encryption scheme which can protect the communication from profilers? Intuitively a cipher obtained from a secure encryption algorithm should not reveal any information regarding the plaintext which was used to produce the cipher. Hence, a secure encryption algorithm should surely resist attacks by profilers. But, as the goal of a profiler is only to classify the messages, it is possible that an encryption algorithm which provides security in a weaker sense would be enough to resist profilers. We explore in this direction and try to fix the appropriate security definition of an encryption scheme which would provide security against profilers.

Using any encryption scheme involves the complicated machinery of key exchange (for symmetric encryption) or a public key infrastructure (for asymmetric encryption). When the goal is just to protect information against profilers the heavy machinery of key exchange or public key infrastructure may be unnecessary. Keeping in mind security against profilers we propose a new protocol which does not require explicit key exchange. To do this we use the notion of CAPTCHAs, which are programs that can distinguish between humans and machines by automated Turing tests which are easy for humans to pass but difficult for any machine. The use of CAPTCHAs makes our protocol secure from non-human profilers, but the protocol is still vulnerable to human adversaries. In the context that we see the activity of profiling, it would be only profitable if a large number of users can be profiled and this goal is infeasible if human profilers are employed for the task.

To our knowledge the only prior works on the issue of securing email communication from profilers have been reported in [5]. In [5] it was pointed out that an encryption scheme secure against profilers can be much weaker than normal encryption algorithms, and thus using a normal encryption algorithm can be an overkill. The solution in [5] hides the semantic of the plaintext by converting an English text into another English text with the help of a key. In their protocol also they do not need explicit key exchange or a public key infrastructure. The key is derived from the email header by using a hash function with a specific property. The hash function they use is a "slow one-way hash function", which was first proposed in [3]. Such hash functions are difficult to compute, i.e., may take a few seconds to get computed and are hard to invert. This high computational cost for the hash function prevents a profiler to derive the key for a large number of messages. Our method is fundamentally different from [5] in its use of CAPTCHAs. Slow hash functions which were proposed long ago have not seen much use, and its suitability is not well tested. But CAPTCHAs are ubiquitous in today's world and had been used successfully in diverse applications. Also, our work presents a theoretical analysis of the problem, and provides the security definitions which to our knowledge is new to the literature.

2 Preliminaries

2.1 Notations

The set of all n bit strings would be denoted by $\{0,1\}^n$. For a string x, $|x|$ will denote the length of x and for a finite set A, $|A|$ would denote the cardinality of A. For a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ will denote x to be an element selected uniformly at random from $\mathcal{S}$. In what follows, by an adversary we shall mean a probabilistic algorithm which outputs an integer or a bit. $\mathcal{A}(x,y) \Rightarrow b$, will denote the fact that an adversary $\mathcal{A}$ given inputs x, y outputs b. In general an adversary would have other sorts of interactions, maybe with other adversaries and/or algorithms before it outputs, these would be clear from the context. In what follows by $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ would denote an encryption scheme with $\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$ as the key space, message space and cipher space respectively. For $m \in \mathcal{M}$ and $k \in \mathcal{K}$ we shall usually write $E_k(m)$ instead of $E(k,m)$.

2.2 Indistinguishability in the Presence of an Eavesdropper

Security of encryption schemes is best defined in terms of indistinguishability. Here we consider indistinguishability in presence of an eavesdropping adversary. This security notion, would be called as IND-EAV security. It considers that an adversary chooses a pair of plaintext messages and then ask for the encryption of those messages. The challenger provides the adversary with the encryption of one of the messages chosen by the adversary. The adversary is considered to be successful if it can correctly guess which message of its choice was encrypted. More formally, to define the security of an encryption algorithm $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$, we consider the interaction of an adversary $\mathcal{A}$ with a challenger in the experiment below:

Experiment Exp-IND-EAV$^{\mathcal{A}}$

1. The challenger selects K uniformly at random from $\mathcal{K}$.
2. The adversary $\mathcal{A}$ selects two messages $m_0, m_1 \in \mathcal{M}$, such that $|m_0| = |m_1|$.
3. The challenger selects a bit b uniformly at random from $\{0, 1\}$, and returns $c \leftarrow E_K(m_b)$ to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ outputs a bit b'.
5. If $b = b'$ output 1 else output 0.

Definition 1. *Let $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ be an encryption scheme. The* IND-EAV *advantage of an adversary $\mathcal{A}$ in breaking E is defined as*

$$\mathbf{Adv}_E^{\text{ind-eav}}(\mathcal{A}) = \Pr[\text{Exp-IND-EAV}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}.$$

Moreover, E is (ϵ, t) IND-EAV *secure if for all adversaries $\mathcal{A}$ running for time at most t, $\mathbf{Adv}_E^{\text{ind-eav}}(\mathcal{A}) \leq \epsilon$.* $\diamondsuit$

The IND-EAV security as defined above is used only for one time encryption and it is different from the most used security notion for symmetric encryption which is indistinguishability under chosen plaintext attack (IND-CPA). In an IND-CPA attack the adversary is given access to the encryption oracle and thus can consult this oracle before it chooses the messages, and has the option of asking encryption of multiple pairs of messages before it outputs. IND-EAV notion is strictly weaker than the IND-CPA notion of security. All IND-CPA secure encryption schemes are also IND-EAV secure.

A related notion of security is that of semantic security. Informally a symmetric encryption scheme is called semantically secure if an adversary is unable to compute any function on the plaintext given a ciphertext.

Definition 2. *Let $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ be an encryption scheme. E is called (ϵ, t)* SEM-EAV *secure, if for all functions f and for all adversaries running for time at most t*

$$|\Pr[\mathcal{A}(E_K(x)) \Rightarrow f(x)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)]| \leq \epsilon \tag{1}$$

where the running time of $\mathcal{A}'$ is polynomially related to t, and x is chosen uniformly at random from $\mathcal{M}$. $\diamondsuit$

Note, in the above definition, by $\mathcal{A}'(.)$ we mean that the adversary is given no input, i.e., $\mathcal{A}'$ is trying to predict $f(x)$ without seeing $E_K(x)$. And in the second term of Equation (1) the maximum is taken over all adversaries $\mathcal{A}'$ which runs for time at most $poly(t)$, for some polynomial $poly()$. Thus, if E is EAV-SEM secure then no adversary can do better in predicting $f(x)$ from $E_K(x)$ than an adversary who does so without seeing $E_K(x)$. It is well known that IND-EAV security implies SEM-EAV security (for example see Claim 3.11 in [9]).

2.3 Captcha

A CAPTCHA is a computer program designed to differentiate a human being from a computer. The fundamental ideas for such a program were first proposed in an unpublished paper [10] and then these ideas were formalized in [15], where the name CAPTCHA was first proposed. CAPTCHA stands for *Completely Automated Public Turing test to tell Computers and Humans Apart*. In fact, a CAPTCHA is a test which is easy to pass by a human user but hard to pass by a machine. One of the most common CAPTCHAs are distorted images of short strings. For a human it is generally very easy to recover the original string from the distorted image, but it is difficult for state of the art character recognition algorithms to recover the original string from the distorted image. Other types of CAPTCHAs which depend on problems of speech recognition, object detection, classification etc. have also been developed.

Recently CAPTCHAs have been used in many different scenarios for identification of humans, like in chat rooms, online polls etc. Also they can be used to prevent dictionary attacks on the password based systems [12], and more recently for key establishment [4].

A CAPTCHA is a randomized algorithm G, which given a input string from a set of strings STR produces the CAPTCHA $G(x)$. A CAPTCHA G is called (α, β) secure, if for any human or legitimate solver S

$$\Pr[x \stackrel{\$}{\leftarrow} \mathsf{STR} : S(G(x)) \Rightarrow x] \geq \alpha,$$

and for any efficient machine C

$$\Pr[x \stackrel{\$}{\leftarrow} \mathsf{STR} : C(G(x)) \Rightarrow x] \leq \beta,$$

For a CAPTCHA to be secure it is required that there is a large gap between α and β. In section 4, we will propose an alternative security definition for CAPTCHAs.

2.4 Secret Sharing Schemes

A secret sharing scheme is a method designed to share a secret between a group of participants. These schemes were first proposed by Shamir in 1979 [13]. Although there have been improvements to these kind of schemes, here we will use the basic construction due to Shamir. In a (u, w) threshold secret sharing scheme a secret K is divided into w pieces called *shares*. These w shares are given to w participants. To recover the secret, at least $u \leq w$ of the w shares are required. And it is not possible to recover the secret with less than u shares.

We describe a specific construction proposed by Shamir. To construct a (u, w) secret sharing scheme we need a prime $p \geq w + 1$ and the operations take place in the field $\mathbb{Z}_p$. The procedure for splitting a secret K into w parts is depicted in the algorithm below:

$\mathsf{SHARE}_{u,w}(K)^p$

1. Choose w distinct, non-zero elements of $\mathbb{Z}_p$, denote them as x_i, $1 \leq i \leq w$.
2. Choose $u-1$ elements of $\mathbb{Z}_p$ independently at random. Denote them as $a_1, \ldots, a_{u-1}$.
3. Let, $a(x) = K + \sum_{j=1}^{u-1} a_j x^j \bmod p$, and $y_i = a(x_i)$, $1 \leq i \leq w$.
4. Output $\mathcal{S} = \{(x_1, y_1), \ldots, (x_w, y_w)\}$ as the set of w shares.

The secret K can be easily recovered using any $B \subset \mathcal{S}$ such that $|B| \geq u$, but if $|B| < u$ then K cannot be recovered. To see this, observe that the polynomial used in step 3 to compute the y_is is a $u-1$ degree polynomial. Thus using u pairs of the type (x_i, y_i) one can generate u linear equations, each of the type $y_i = K + a_1 x_i + \cdots a_{u-1} x_i^{u-1}$. Using these equations the value of K can be found. It can be shown that this set of u equations would always have a unique solution.

3 Profiling Adversaries

Let $\mathcal{M}$ be a message space and $\mathcal{P} = \{1, 2, \ldots, k\}$ be a set of labels for different possible profiles. We assume that each message x in $\mathcal{M}$ can be labeled by a unique $j \in \mathcal{P}$. Thus, there exists a function $f : \mathcal{M} \to \mathcal{P}$, which assigns a label to each message in the message space. In other words, we can assume that the message space can be partitioned into disjoint subsets as $\mathcal{M} = M_1 \cup M_2 \cup \cdots \cup M_k$ and for every $x \in \mathcal{M}$, $f(x) = i$ if and only if $x \in M_i$.

We call f as the profiling function or a classifier. Thus, in this setting we are assuming that each message in the message space $\mathcal{M}$ represents some profile, and messages in $M_i (1 \leq i \leq k)$ correspond to the profile i. The function f is a classifier which given a message can classify it into one of the profiles. We also assume that the function f is efficiently computable for every $x \in \mathcal{M}$, in particular, we assume that for any $x \in \mathcal{M}$, $f(x)$ can be computed in time at most μ, where μ is a constant.

The function f is public, thus given $x \in \mathcal{M}$ any adversary can efficiently compute $f(x)$. We want to define security for an encryption scheme which is secure against profiling adversaries, i.e., we want that when a message from $\mathcal{M}$ is encrypted using the encryption algorithm no efficient adversary would be able to profile it.

3.1 PROF-EAV Security

Here we propose a definition for encryption schemes secure against profiling adversaries.

Definition 3. **[PROF-EAV security]** *Let $\mathcal{M}$ be a message space and $f : \mathcal{M} \to \mathcal{P}$ be a profiling function. Let $E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ be an encryption algorithm. We define the advantage of an adversary $\mathcal{A}$ in the* PROF-EAV *(read profiling under eavesdropping) sense in breaking E as*

$$\begin{aligned} &\mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{A}) \\ &= \Pr[\mathcal{A}(E_K(x)) \Rightarrow f(x)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)], \end{aligned} \tag{2}$$

where $K \xleftarrow{\$} \mathcal{K}$, $x \xleftarrow{\$} \mathcal{M}$ and $\mathcal{A}'$ is an adversary whose running time is a polynomial of the running time of $\mathcal{A}$. An encryption algorithm $E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ is called (ϵ, t) PROF-EAV *secure for a given profiling function f, if for all adversaries $\mathcal{A}$ running in time at most t, $\mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{A}) \leq \epsilon$.* ♢

In the definition above, we want to capture the notion that for a PROF-EAV secure encryption scheme, an adversary $\mathcal{A}$ trying to find the profile of a message seeing its cipher cannot do much better than the best adversary $\mathcal{A}'$, who tries to guess the profile without seeing the ciphertext.

This definition is in accordance with the definition of semantic security as discussed in Section 2.2. Recall that an encryption scheme is called semantically secure if no adversary can efficiently compute *any* function of the plaintext given its ciphertext. But in the PROF-EAV definition we are interested only with a specific function f. Thus, PROF-EAV security is strictly weaker than semantic security. Semantic security trivially implies PROF-EAV security but PROF-EAV security does not imply IND-EAV security, we give a concrete example to illustrate this.

Example 1. Let $\mathcal{M} = \{0,1\}^n = M_1 \cup M_2$ be a message space, where

$$M_1 = \{x \in \mathcal{M} : \text{first bit of } x \text{ is } 0\},$$

and $M_2 = \mathcal{M} \setminus M_1$, and f be the profiling function such that $f(x) = i$ iff $x \in M_i$. Let E^{one} be an encryption scheme which uses a one bit key k (chosen uniformly from $\{0,1\}$) and given a message $x \in \mathcal{M}$ it xors k with the first bit of x. It is easy to see that an adversary trying to guess the profile of a message x given $E^{\text{one}}_k(x)$ cannot do better than with probability half, and this success probability can be achieved even without seeing the ciphertext, as here $|M_1| = |M_2|$. Hence E^{one} is PROF-EAV secure, but trivially not secure in the IND-EAV sense.

4 Encryption Protocol Secure Against Profiling Adversaries

In this section we describe a complete protocol which would be secure from profiling adversaries. As mentioned in the introduction here we care about adversaries who are not humans. Our motivation is to prevent communications getting profiled in large scale mechanically. The protocol is not secure from human adversaries, and we do not care much about that as we hope that it would be economically and otherwise infeasible to employ a human for large scale profiling.

The protocol $\mathbb{P}$ consists of the following entities:

- The message space $\mathcal{M}$, the cipher space $\mathcal{C}$.
- The set of profiles $\mathcal{P}$ and the profiling function f associated with $\mathcal{M}$.
- A set STR which consists of short strings over a specified alphabet.
- An encryption scheme $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$.
- A hash function $H : \mathsf{STR} \to \mathcal{K}$.
- A CAPTCHA generator G which takes inputs from STR.

Given a message $x \in \mathcal{M}$, $\mathbb{P}$ produces a ciphertext as shown in Figure 1. In the

Protocol $\mathbb{P}(x)$
1. $k \xleftarrow{\$} \mathsf{STR}$;
2. $k' \leftarrow G(k)$;
3. $K \leftarrow H(k)$;
4. $c \leftarrow E_K(x)$;
5. **return** (c, k')

Fig. 1. The protocol $\mathbb{P}$.

protocol as described in Figure 1, k, an element of STR is hashed to form the key K and k is also converted into a CAPTCHA and transmitted along with the ciphertext. The only input to $\mathbb{P}$ is the message and the key generation is embedded in the protocol. It resembles the scenario of hybrid encryption [1], which consists of two mechanisms called key encapsulation and data encapsulation where an encrypted version of the key is also transmitted along with the cipher. For a human decryption is easy, as given a ciphertext (c, k') a human user can recover k from k' by solving the CAPTCHA and thus compute $E^{-1}_{H(k)}(c)$ to decipher.

4.1 Security of $\mathbb{P}$

The security of a protocol $\mathbb{P}$ against profilers is defined in the same way as in Definition 3.

Definition 4. *The advantage of an adversary attacking protocol $\mathbb{P}$ is defined as*

$$\begin{aligned}&\mathbf{Adv}^{\mathrm{prof}}_{\mathbb{P},f}(\mathcal{A})\\ &= \Pr[\mathcal{A}(\mathbb{P}(x)) \Rightarrow f(x)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)],\end{aligned}$$

where $x \xleftarrow{\$} \mathcal{M}$ *and* $\mathcal{A}'$ *is an adversary whose running time is a polynomial of the running time of* $\mathcal{A}$*. Additionally* $\mathbb{P}$ *is called* (ϵ, t) *secure in the PROF sense if for all adversaries running in time at most* t*,* $\mathbf{Adv}_{\mathbb{P}}^{\mathrm{prof}}(\mathcal{A}) < \epsilon$*.* $\diamondsuit$

The above definition is different from Definition 3 by the fact that it does not mention the key explicitly, as key generation is embedded in the protocol itself. To prove that $\mathbb{P}$ is secure in the PROF sense we need an assumption regarding the CAPTCHA G and the hash function H. We state this next.

Definition 5. [The Hash-Captcha assumption] *Let* G *be a CAPTCHA generator, let* ℓ *be a number, let* $H : \mathsf{STR} \rightarrow \{0,1\}^{\ell}$ *be a hash function, and let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *in violating the Hash-Captcha assumption as*

$$\begin{aligned}\mathbf{Adv}_{G,H}^{\mathrm{hc}}(\mathcal{A}) = \Pr[x \xleftarrow{\$} \mathsf{STR} : \mathcal{A}(G(x), H(x)) \Rightarrow 1] \\ - \Pr[x \xleftarrow{\$} \mathsf{STR}, z \xleftarrow{\$} \{0,1\}^{\ell} : \mathcal{A}(G(x), z) \Rightarrow 1].\end{aligned}$$

Moreover, (G, H) *is called* (ϵ, t) HC *secure if for all adversaries* $\mathcal{A}$ *running in time at most* t*,* $\mathbf{Adv}_{G,H}^{\mathrm{hc}}(\mathcal{A}) \leq \epsilon$*.* $\diamondsuit$

This definition says that the pair formed by a CAPTCHA generator G and a hash function H is secure, if an adversary $\mathcal{A}$ is unable to distinguish between a $(G(x), H(x))$, where x is some string, and $(G(x), z)$, where z is a random string. This security notion of a CAPTCHA inspired by the notion of *indistinguishability* is quite different from the (α, β) security notion as described in Section 2.3. Here the adversary has some more information regarding x through the value $H(x)$. If the adversary can efficiently solve the CAPTCHA G then it can break (G, H) in the HC sense irrespective of the hash function. Given the CAPTCHA is secure, i.e., no efficient adversary can find x from $G(x)$ still an adversary may be able to distinguish $H(x)$ from a string randomly selected from the range of H.

If we consider a keyed family of hash functions $\mathcal{H} = \{H_\ell\}_{\ell \in \mathcal{L}}$, such that for every $\ell \in \mathcal{L}$, $H_\ell : \mathcal{D} \rightarrow \mathcal{R}$ for some sets $\mathcal{D}$ and $\mathcal{R}$. Then $\mathcal{H}$ is called a entropy smoothing family if for any efficient adversary it is difficult to distinguish between $(\ell, H_\ell(x))$ and (ℓ, z), where ℓ, x, z are selected uniformly at random from $\mathcal{L}$, $\mathcal{D}$ and $\mathcal{R}$ respectively. An entropy smoothing hash along with a secure captcha can resist HC attacks. Entropy smoothing hashes can be constructed from universal hash functions using the left over hash lemma [7], but the parameter sizes which would be required for such provable guarantees can be prohibitive. We believe that using ad-hoc cryptographic hashes like the ones from the SHA family can provide the same security. In our definition we do not use a keyed family of hash functions, but such a family can be easily used in the protocol $\mathbb{P}$, and in that case the hash key will also be a part of the ciphertext.

With these discussions we are now ready to state the theorem about security of $\mathbb{P}$.

Theorem 1. *Let $\mathbb{P}$ be a protocol as in Figure 1 and $\mathcal{A}$ is an adversary attacking $\mathbb{P}$ in the* PROF *sense. Then there exist adversaries $\mathcal{B}$ and $\mathcal{B}'$ such that*

$$\mathbf{Adv}^{\text{prof}}_{\mathbb{P},f}(\mathcal{A}) \leq \mathbf{Adv}^{\text{hc}}_{G,H}(\mathcal{B}) + \mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{B}').$$

And, if $\mathcal{A}$ runs for time t, both $\mathcal{B}$ and $\mathcal{B}'$ runs for time $O(t)$.

Proof. Let $\mathcal{A}$ be an adversary attacking the protocol $\mathbb{P}$ in Figure 1. We construct an adversary $\mathcal{B}$ attacking the hash-captcha, using $\mathcal{A}$ as follows.

Adversary $\mathcal{B}(G(k), z)$
1. $x \xleftarrow{\$} \mathcal{M}$;
2. Send $(E_z(x), G(k))$ to $\mathcal{A}$;
3. $\mathcal{A}$ returns j;
4. **if** $f(x) = j$;
5. **return** 1;
6. **else return** 0;

As $\mathcal{B}$ is an adversary attacking the hash-captcha assumption, hence there are two possibilities regarding the input $(G(k), z)$ of $\mathcal{B}$, z can either be $H(k)$ or a uniform random element in $\mathcal{K}$, and the goal of $\mathcal{B}$ is to distinguish between these two possibilities.

Considering the first possibility that z is $H(k)$, the way the adversary $\mathcal{B}$ is defined, $\mathcal{A}$ gets a valid encryption of the message x (which is a random element in the message space) according to the protocol $\mathbb{P}$. Hence we have

$$\begin{aligned} &\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{B}(G(k), H(k)) \Rightarrow 1] \\ &= \Pr[k \xleftarrow{\$} \mathcal{K}, x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_{H(k)}(x), G(k)) \Rightarrow f(x)] \\ &= \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(\mathbb{P}(x)) \Rightarrow f(x)]. \end{aligned} \tag{3}$$

Similarly, for the second possibility, i.e., when the input z to $\mathcal{B}$ is an element chosen uniformly at random from $\mathcal{K}$, we have

$$\begin{aligned} &\Pr[k, K \xleftarrow{\$} \mathcal{K} : \mathcal{B}(G(k), K) \Rightarrow 1] \\ &= \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_K(x), G(k)) \Rightarrow f(x)]. \end{aligned} \tag{4}$$

In Equation (4), k and K are chosen independently uniformly at random from K. Thus, the adversary $\mathcal{A}$ has as input $E_K(x)$ and $G(k)$, where k is independent of K, thus $G(k)$ carries no information about K. Hence $\mathcal{A}$ cannot do better than some PROF-EAV adversary $\mathcal{B}'$ who has only $E_K(x)$ as its input, and runs for same time as that of $\mathcal{A}$. Thus

$$\begin{aligned} &\Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_K(x), G(k)) \Rightarrow f(x)] \\ &\leq \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{B}'(E_K(x)) \Rightarrow f(x)] \end{aligned} \tag{5}$$

From definition of PROF-EAV advantage of $\mathcal{B}'$ we have

$$\begin{aligned}&\Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{B}'(E_K(x)) = f(x)]\\ &= \mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{B}') + \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)]\end{aligned} \quad (6)$$

Thus, using Equations (4), (5) and (6) we have

$$\begin{aligned}&\Pr[k, K \xleftarrow{\$} \mathcal{K} : \mathcal{B}(G(k), K) \Rightarrow 1]\\ &\leq \mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{B}') + \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)]\end{aligned} \quad (7)$$

Finally, from Equations (3) and (7) and Definitions 5 and 4 we have

$$\mathbf{Adv}^{\text{prof}}_{\mathbb{P},f}(\mathcal{A}) \leq \mathbf{Adv}^{\text{hc}}_{G,H}(\mathcal{B}) + \mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{B}'),$$

as desired. □

Some remarks about security of $\mathbb{P}$: We defined the security of the protocol $\mathbb{P}$ for only a fixed profiling function f, but note that we can modify the definition for any arbitrary function f which would give us a security definition equivalent to SEM-EAV (discussed in Section 2.2). If the encryption algorithm E used within the protocol is SEM-EAV secure then using the same proof we can obtain SEM-EAV security for $\mathbb{P}$.

5 A Practical Instantiation

A very common problem using CAPTCHAs is that sometimes even humans may fail to solve them. As in the protocol $\mathbb{P}$ if a human user fails to solve the CAPTCHA then (s)he will not be able to decipher and there is no way to repeat the test (as is done in normal CAPTCHA usage), hence this stands as a serious weakness of the proposed protocol $\mathbb{P}$. A solution to this problem can be attempted by providing some redundancy in the CAPTCHAs so that a valid user can have more chance in solving the CAPTCHA. As a solution we propose that the initial string k chosen by the protocol be broken into w shares such that with t or more of the shares would be enough to generate k. These w shares are converted into CAPTCHAs and sent along with the ciphertext. To incorporate this idea we changed the initial protocol $\mathbb{P}$ to $\mathbb{P}'$. The protocol $\mathbb{P}'$ is a specific instantiation, thus before we describe the protocol we fix some details of its components, in particular for $\mathbb{P}'$ we would require an encoding mechanism ENCD which we discuss first.

Let $\mathsf{AL} = \{A, B, \ldots, Z\} \cup \{a, b, \ldots, z\} \cup \{0, 1, \ldots, 9\} \cup \{+, /\}$, thus making $|\mathsf{AL}| = 64$. We define an arbitrary (but fixed) bijection $\rho : \mathsf{AL} \to \{0, 1, \ldots, 63\}$, and for any $\sigma \in \mathsf{AL}$ and $n \geq 6$, $\mathsf{bin}_n(\sigma)$ will denote the n bit binary representation of $\rho(\sigma)$. Note that for all $\sigma \in \mathsf{AL}$, at most 6 bits are required to represent $\rho(\sigma)$. If ψ is a binary string, then let $\mathsf{toInt}(\psi)$ be the positive integer corresponding to ψ, similarly for a positive integer $v < 2^n$, $\mathsf{toBin}_n(v)$ denotes the n bit binary representation of v. We fix a positive integer m and let STR be the set of all m character strings over the alphabet AL. Let p be the smallest prime greater than 2^{6m} and let $d = p - 2^{6m}$. Let $\mathsf{ENCD} : \mathsf{STR} \times \{0, 1, ..., d\} \rightarrow \mathbb{Z}_p$ be defined as follows

$\mathsf{ENCD}(s, \lambda)$
1. Parse s as $\sigma_0||\sigma_1||\ldots||\sigma_m$, where each $\sigma_i \in \mathsf{AL}$;
2. $\psi \leftarrow \mathsf{bin}_6(\sigma_0)||\ldots||\mathsf{bin}_6(\sigma_m)$;
3. $v \leftarrow \mathsf{toInt}(\psi)$;
4. **return** $v + \lambda$;

And let $\mathsf{ENCD}^{-1} : \mathbb{Z}_p \rightarrow \mathsf{STR} \times \{0, 1, \ldots, d\}$ be defined as

$\mathsf{ENCD}^{-1}(y)$
1. **if** $y \geq 2^{6m}$,
2. $\quad \lambda \leftarrow y - 2^{6m} + 1$;
3. $\quad y \leftarrow 2^{6m} - 1$;
4. **else** $\lambda \leftarrow 0$;
5. $z \leftarrow \mathsf{toBin}_{6m}(y)$;
6. Parse z as $z_0||z_1||\ldots||z_m$, where $|z_i| = 6$;
7. $s \leftarrow \rho^{-1}(\mathsf{toInt}(z_0))||\ldots||\rho^{-1}(\mathsf{toInt}(z_m))$;
8. **return** (s, λ);

The modified protocol $\mathbb{P}'$ is shown in Figure 2. It uses the encoding function ENCD and the secret sharing scheme as depicted in Section 2.4. For $\mathbb{P}'$ we assume that STR contains all m character strings over the alphabet AL, and p is the smallest prime greater than 2^{6m}, these can be considered the fixed and public parameters for $\mathbb{P}'$. The encoding mechanism is specifically designed to convert a string in STR to an element in $\mathbb{Z}_p$ so that Shamir's secret sharing can be suitably used.

To decrypt a cipher produced by $\mathbb{P}'$ a human user must solve at least some u of w CAPTCHAs. Using these u solutions together with x_i, k can be recovered. A specific recommendation for SHARE can be Shamir (2,5)-threshold scheme. Thus the user would have much flexibility on solving the CAPTCHAs.

5.1 Security of $\mathbb{P}'$

The security of $\mathbb{P}'$ can be easily proved in the sense of Definition 4 in a similar way as we prove Theorem 1 if we make a new assumption regarding the CAPTCHA as follows :

Definition 6. [The Hash-MultiCaptcha assumption] *Let G be a CAPTCHA generator, let ℓ be a number, let $H : \mathsf{STR} \rightarrow \{0, 1\}^{\ell}$ be a hash function, and let $\mathcal{A}$ be an adversary. Also, let $x = g(x_1, \ldots x_w)$ be such that if at least u out of w*

Protocol $\mathbb{P}'(x)$

1. $k \xleftarrow{\$} \mathsf{STR}$;
2. $k' \leftarrow \mathsf{ENCD}(k, 0)$;
3. $\{(x_1, k'_1), \ldots, (x_w, k'_w)\} \leftarrow \mathsf{SHARE}^p_{u,w}(k')$;
4. **for** $i = 1$ **to** w;
5. $\quad (k_i, \lambda_i) \leftarrow \mathsf{ENCD}^{-1}(k'_i)$;
6. $\quad c_i \leftarrow G(k_i)$;
7. **end for**
8. $K \leftarrow H(k)$;
9. $C \leftarrow E_K(x)$;
10. **return** $[C, \{(x_1, c_1, \lambda_1), \ldots, (x_w, c_w, \lambda_w)\}]$

Fig. 2. The protocol $\mathbb{P}'$ which uses a secret-sharing scheme.

of $x_1, \ldots, x_w$ are known then x can be recovered. We define the advantage of $\mathcal{A}$ in violating the Hash-MultiCaptcha assumption as

$$\begin{aligned}\mathbf{Adv}^{\mathrm{mhc}}_{G,H}(\mathcal{A}) = \Pr[&\mathcal{A}(G(x_1), \ldots G(x_w), H(x)) \Rightarrow 1] \\ &- \Pr[z \xleftarrow{\$} \{0,1\}^{\ell} : \mathcal{A}(G(x_1), \ldots G(x_w), z) \Rightarrow 1].\end{aligned}$$

where $x \xleftarrow{\$} \mathsf{STR}$. Moreover, (G, H) is called (ϵ, t) HMC secure if for all adversaries $\mathcal{A}$ running in time at most t, $\mathbf{Adv}^{\mathrm{mhc}}_{G,H}(\mathcal{A}) \leq \epsilon$. ♢

As in the definition of Hash-Captcha assumption, in this definition if the adversary can efficiently solve at least u of w CAPTCHAs, then it can break (G, H) in the HMC sense irrespective of the hash function. If this assumption is true, then we can show the security of protocol $\mathbb{P}'$ just as we did for protocol $\mathbb{P}$.

A CAPTCHA is an example of a weakly-verifiable puzzle [2], since a legitimate solver S may not be able to verify the correctness of its answer. For this kind of puzzles, it has been proved [6] that if it is difficult for an attacker to solve a weakly-verifiable puzzle P, then trying to solve multiple instances of a puzzle in parallel is harder. Most recently, Jutla found a better bound to show how hard it is for an attacker to solve multiple instances of weakly-verifiable puzzles [8]. The next theorem is based on the main theorem proposed by Jutla, but it has been adapted to CAPTCHAS, which are of our interest in this work.

Theorem 2. *Let G be a CAPTCHA generator which is (α, β) secure. Let $k \in \mathbb{N}$, $\delta = 1 - \beta$ and γ $(0 < \gamma < 1)$ be arbitrary. Let $\mathcal{A}$ be an arbitrary polynomial time adversary, which is given as input k CAPTCHAs $(G(x_1), \ldots, G(x_k))$ and outputs a set X of solutions of the k CAPTCHAs. If $\mathsf{InCorr}(X)$ denotes the number of incorrect solutions in X, then*

$$\Pr[\mathsf{InCorr}(X) < (1 - \gamma)\delta k] < e^{-(1-\gamma)\gamma^2 \delta k/2}$$

This theorem establishes that for any adversary if the probability of failure in solving a CAPTCHA is at least δ, then the probability of failing on less than $(1-\gamma)\delta k$ out of k puzzles, is at most $e^{-(1-\gamma)\gamma^2\delta k/2}$.

Based on this fact, it may be possible to show that for any arbitrary adversary $\mathcal{A}$ attacking the HMC assumption, there exists a HC adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathrm{mhc}}_{G,H}(\mathcal{A}) < \mathbf{Adv}^{\mathrm{hc}}_{G,H}(\mathcal{B})$. This would imply that the HC assumption implies the HMC assumption. But, for now we are not sure whether such a result holds.

5.2 Discussions

- **About the encryption scheme:** In this work we have not said anything about the encryption scheme to be used in the protocol. We only said that we require our encryption scheme to be PROF-EAV secure and any IND-EAV secure encryption scheme can provide such security. Thus most symmetric encryption schemes which are usually in use like CBC mode, counter mode etc. (which provide security in the IND-CPA sense) can be used for the encryption function E in $\mathbb{P}'$. A more efficient scheme which provides security only in the PROF-EAV sense would be much interesting, we would like to explore in this direction.
- **Key sizes:** Another important thing to consider is that the effective size of a key for the protocol is dictated by the parameter m, i.e., the size of each string in STR. This value cannot be made arbitrarily large as solving big CAPTCHAs for human beings may be tiresome, a usual CAPTCHA length is five to eight characters. If we use eight character strings from the alphabet AL then the effective size of the key space would be 2^{48}. Increasing the alphabet size is also not feasible as we need un-ambiguous printable characters to make CAPTCHAs. Thus, the key space is not sufficiently large for a modern cryptographic application, but for the application which we have in mind this may be sufficient, as we do not expect that a profiler would be ready to use so much computational resource for profiling a single message.

6 Final Remarks

In this paper we did a theoretical analysis of profiling adversaries and ultimately described a protocol which is secure against profiling adversaries. Our protocol does not require any key exchange or public key infrastructure and uses CAPTCHAs and secret sharing schemes in a novel way.

Encryption may not be the only way to protect a user from profilers. As profilers can use many different techniques which cannot be stopped using encryption. For example it is possible to track the web usage of a specific user and profile him/her on that basis. Here (probably) encryption has no role to play, or at least cannot be used in the way we propose in our protocol. Anonymity is probably the correct direction to explore in solving such problems. Also, as user profiling is a big business, and some think that the free content in the web is only possible due to online advertisements, so putting a total end to user profiling may not be desirable. So there have been current attempts to develop systems which would allow targeted advertisements without compromising user security [14]. These issues are not covered in our current work.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.
2. R. Canetti, S. Halevi, and M. Steiner. Hardness amplification of weakly verifiable puzzles. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2005.
3. C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.
4. S. Dziembowski. How to pair with a human. In J. A. Garay and R. D. Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 200–218. Springer, 2010.
5. P. Golle and A. Farahat. Defending email communication against profiling attacks. In V. Atluri, P. F. Syverson, and S. D. C. di Vimercati, editors, *WPES*, pages 39–40. ACM, 2004.
6. R. Impagliazzo, R. Jaiswal, and V. Kabanets. Chernoff-type direct product theorems. *J. Cryptology*, 22(1):75–92, 2009.
7. R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *FOCS*, pages 248–253. IEEE, 1989.
8. C. S. Jutla. Almost optimal bounds for direct product threshold theorem. In D. Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2010.
9. J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2008.
10. M. Naor. Verification of a human in the loop or identification via the turing test, 1997. `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.pdf`.
11. NYT. Congress begins deep packet inspection of internet providers, 2009. `http://bits.blogs.nytimes.com/2009/04/24/congress-begins-deep-packet-inspection-of-internet-providers/`.
12. B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In V. Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 161–170. ACM, 2002.
13. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
14. V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Privacy preserving targeted advertising. In *Proceedings of annual network and distributed systems security symposium*, 2010. `http://www.isoc.org/isoc/conferences/ndss/10/pdf/05.pdf`.
15. L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.