How *to Prevent* to Delegate Authentication

Mohsen Alimomeni(✉) and Reihaneh Safavi-Naini

University of Calgary, Calgary, AB, Canada
{malimome,rei}@ucalgary.ca

Abstract. We consider *delegation attack* in authentication systems in which a credential holder shares their credentials with a third party that we call *helper*, to allow them to use their account. We motivate this problem and propose a model for non-delegatable authentication and a novel authentication system, based on behavioural biometrics, that achieves non-delegatability. Our main observation is that a user's behaviour in complex activities such as playing a computer game, provides an imprint of many of their personal traits in the form of measurable features, that can be used to identify them. Carefully selected features will be "hard" to pass on to others, hence providing non-delegatability. As a proof of concept we designed and implemented a computer game (a complex activity), and used the feature points in the game play to construct a user model for authentication. We describe our implementation and experiments to evaluate correctness, security and non-delegatability. Compared to using traditional biometrics, the system enhances user privacy because the user model is with respect to an activity and do not have direct relation to the user's identifying information. We discuss our results and deployment of the system in practice, and propose directions for future research.

1 Introduction

We consider the problem of credential sharing, where a user wants to share their credential with a third party with the goal of bypassing the system security. We refer to this as *delegation attack*. The problem naturally arises in authentication systems (e.g. online subscription systems) where users have incentives to share their credentials and let a third party use their privileges, or assume their roles. Traditional authentication systems do not provide protection against this attack. Authentication systems use credentials such as, what a user knows (e.g. passwords, secret keys), what a user has (e.g. tokens, cards), and what a user is (biometric) to ensure correct identity claims. They may also use user *attributes* such as their expected location or distance from the verifier, to provide stronger security guarantees. In all cases security of an authentication system is primarily against an *outside attacker* who, without having access to the user credentials, tries to impersonate them.

We consider a scenario that a user actively shares their credential. In this case security of all known traditional authentication systems will be severely compromised. Systems that rely on secret keys (or passwords) and tokens cannot

B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 477–499, 2016.
DOI: 10.1007/978-3-319-28865-9_26

provide any security guarantee. Biometric systems that rely on the user's unique characteristics (e.g. fingerprint, voiceprint) may also become insecure if the user is willing to share their biometric templates [Fid13]. Systems that use attributes such as distance of a user to the verifier usually rely on a secret key (symmetric or public key) and cannot guarantee security if the secret key is passed on.

Credential sharing is a well known problem in subscription services such as Netflix [Wor13] and online games [TBB12] and can effectively bypass the security of the subscription system. The problem is widely studied and a range of solutions including trusted hardware and tamper-proof software have been proposed. However solutions that provide sufficient usability for the system (e.g. allowing multiple devices), quickly become ineffective. In corporate world credential sharing is a known problem, commonly used for reasons such as ease of access to documents (e.g. an executive shares their password with their assistants to allow them access). A less studied problem however, is credential sharing by dishonest employees with motivations such as employing "cheap labour" from outside the company to perform one's allocated tasks, or organizing more systematic collusion (e.g. espionage) attacks to provide access to outsiders. The former case has been a real concern of software companies where employees delegate software development tasks to developer sites that offer this service [TH13]. Correct authentication of remote users is also increasingly important due to the wider adoption of work-from-home model, and the need for companies to cater for mobile workforce.

An immediate solution for providing security against credential sharing is to use additional factors such as a hardware token, in the authentication process. Tokens however, although make it harder for users to pass on their credentials, cannot protect against credential sharing: a software developer [TH13] in the US outsourced their work to a Chinese firm by sending the RSA token that was required for authentication. A second solution is to use biometric based authentication systems. Biometric templates although in general are unique to individuals, in some cases may be recorded and replayed for authentication [Fid13]. However in the above application scenarios, it is perceivable that one will not be willing to share their biometric data because of the permanency and sensitivity of this data. Biometric systems have disadvantages such as the need for extra hardware and deployment cost, in addition to careful management of the collected biometric data throughout the lifetime of the system. Using biometric authentication in corporate environment also introduces privacy concerns for employees who may move from one employer to another, and do not want to leave a biometric trace behind. A third solution is to strengthen password systems using extra behavioural features of users. Existing behavioural authentication systems capture simple users' behaviours such as keyboard typing pattern or mouse dynamics [MR00] and have no real guarantee that these behaviours cannot be taught or transferred to others. Our method can be seen as developing this approach by designing activities that capture complex non-transferable characteristics of users.

1.1 Our Work

Intuitively, to prevent delegation of authentication credential, one must use intrinsic properties of users that are "hard" to pass on to others. Such properties can be grouped into *personality traits*, and *behavioural* and *cognitive* factors. Identifying individuals using their intrinsic properties have been subject of extensive studies in psychology. Trait theory approach to personality promotes the idea that individuals can be identified through their personality traits such as abstractedness, perfectionism and reasoning. Cattell suggests 16 personality factors [Cat57] are sufficient to identify individuals. Human behaviour refers to one's actions and manners in response to stimuli (inputs) that could be internal or external, and conscious or subconscious. Human behaviour has been shown to be effective in distinguishing individuals [BSR+12, MR00]. Cognitive abilities in domains such as language, reasoning, memory, learning and visual perception, as well as higher order abilities such as intelligence, have been measured through well designed experiments and shown to be able to identify individuals [Car93]. We use *personal traits* to refer to both these types of human intrinsic properties when they are, (i) *measurable* in the interactions of users with the environment, and (ii) are relatively *stable*. Stability of a trait intuitively refers to the property that the measurements of the trait correspond to a narrow probability distribution that could be used to differentiate users in a population. Stable traits may change over time. We assume this change can be represented by a (slow) shift over time. Traits may have different levels of transferability. Some traits may be learnt or imitated by training and practice (with different degrees of success). For example, traits related to the user behaviour (personal preferences) can be learnt more easily than skill based traits such as speed of performing an action.

Our work aims to capture *trait related information* of an individual in a complex activity. A measurement in an activity is modelled by a random variable, representing in general, multiple personal traits. The *user profile* consists of these variable, also called *features*. Features are chosen to be non-delegatable in the sense that they are "hard" to be learnt by a helper that is assisted by the user. We call authentication systems built on these profiles, a *Hard to Delegate (HtD)* authentication system. As a proof of concept we designed and implemented a target shooting game to model a complex activity. In an authentication attempt a *challenge* is presented to the user and their response is received. The challenge is a game (in our case a target), and the response is a set of measurements during their game play (in our case, an arrow shot at the target). The response measurements is matched against the stored user profile. To analyze the system, we first give a formal definition of non-delegatability in authentication systems. This is a new security property that captures protection against a user credential sharing. We then use user experiments in small groups to select non-delegatable features, followed by large group experiments for evaluating correct user authentication. We also design and implement special experiments to show that the system provides protection against non-delegatability.

We note that non-delegatability is a strictly stronger security requirement than user impersonation, because the credential holder assists the attacker to succeed in impersonating them and the proposed system is also a new secure authentication system using user game play.

Selecting Features in Activities. A feature in an activity is a measurement that corresponds to a random variable X. This variable is sampled in each run of the activity, producing a *feature point* x. The randomness of the variable is due to the user's *intrinsic randomness* that results from the complex combination of their personal traits. Suitable features to support non-delegatability must be, (i) strongly correlated with stable user traits and stay stable over time and, (ii) be hard to transfer. Selecting such features in our system has been through small group experiments. The experiments (described in Section 5.3) suggests that selecting effective features is a rich direction for future research. An interesting case is *tightly coupled features* that provide strong non-delegatability. These are pairs of features that are negatively correlated, but successful impersonation requires both to be modified in the same direction. For example in our target shooting game, the speed at which a user aims at the target and the error in hitting the target are negatively correlated (i.e. reducing aim time increases error). However to imitate a (skilled) user one needs to reduce aim time and error at the same time.

Applications. Non-delegatable authentication systems can be used in conjunction with traditional password based (or key-based) authentication to provide non-delegatability. Our motivating example was providing security for work from home environment that could pose major threat to the enterprise network. Another important application is providing protection against credential sharing in massively multiplayer online (MMO) games with incentives such as bypassing subscription fees, allowing a more experienced player to play on one's behalf, or hijacking an account [CH07] to take advantage of the user's progress in the game. An important advantage of behavioural authentication system such as the one proposed in this paper is privacy enhancement because of using the behavioural attributes instead of personally identifiable information.

Ethics Approval. The experiments described in this paper involved human subjects. We obtained ethics approval from the Conjoint Faculties Research Ethics Board at the University of Calgary, under the file number 7630. The first author completed a course on ethics, entiled Ethical Conduct for Research Involving Humans Course on Research Ethics (TCPS 2: CORE). All experiments were performed in accordance with these ethics guidelines.

1.2 Related Works

Behavioural biometrics [Rev08] is a relatively new research area. Human computer interaction based biometrics such as those based on keystroke dynamics[MR00] and mouse movement [PB04], have been shown to be effective way of identifying users. In [YG09], authors showed that measuring the player's strategy in a poker game is effective for user verification. Our approach of using feature points that are behaviour based is distinctly different from collecting feature points related to the user strategy as used in [YG09] for the game of poker. This latter type of points are not chosen for non-delegatability and in fact may be delegatable. Alayed et al. [AFN13] used a first person shooter game to distinguish between normal behaviour of the players, and cheating behaviour. The output of their classifier is a binary value, indicating cheating or no cheating.

Implicit memory for authentication was proposed by Denning et al. [DBvDJ11]. Bojinov et al. [BSR+12] used implicit learning to defend against "rubber hose attacks" in authentication. Implicit learning cannot directly prevent delegation attack because a dishonest user may memorize the password during the training phases and later pass it on to the helper. HtD authentication however can achieve the goals of [DBvDJ11] and [BSR+12] without requiring password.

Paper Organization. Section 2, gives a model for HtD property. Section 3 is on behavioural biometric using complex non-debatable features. Section 4, is our proof of concept game, the collected features of users and describe the experimental setup and the results. Section 3.2 is on deployment issues and attacks on HtE games, and cheat-proofing techniques for preventing these attacks.

2 Non-delegatable Authentication

A HtD system has three computational entities, a Server S, a Client C, and a device D with three interfaces DI_1, DI_2 and DI_3, that are used to present a challenge to the user, collect the response from the user, and communicate with the network, respectively. S sends the challenge to C on the device D using DI_1. The user responds using DI_2 that is passed to C, which is finally forwarded to S via DI_3.

2.1 HtD Authentication Systems

We consider a multiparty setting where participants receive inputs and produce outputs. An honest participant follows the protocol and a dishonest one deviates arbitrarily, in all cases using probabilistic polynomial-time (PPT) algorithms. A participant can be a *prover* denoted by $\mathcal{P}$ (also referred to as a *user* U), a *verifier* denoted by $\mathcal{V}$, or an *adversary* denoted by $\mathcal{A}$. The adversary corrupts participants and uses them to defeat security of the system. The verifier V always behaves honestly. A prover however may be corrupted, in which case it is denoted by $\mathcal{P}^*$. A prover $\mathcal{P}$ has a set of attributes some measurable directly (e.g. location, IP), and some indirectly through *imprints* that are obtained during a user activity. These can be *estimated* through random variables that are measured during user activities. The random variables in general take different values in different measurement rounds, following a (slow changing) distribution. For example, the error in hitting the target in a target shooting game, carries user intrinsic attributes such as their skill level in the game play. A prover $\mathcal{P}$ thus is intrinsically probabilistic and its attributes in general can be represented as a vector of random variables[1].

Authentication Protocol. An HtD authentication protocol is a two party protocol between two interactive PPT algorithms, a trusted verifier $\mathcal{V}$ and a prover $\mathcal{P}$. We also use V and P to refer to the verifier and the prover, respectively.

A *protocol run (instance)* between $\mathcal{P}$ and $\mathcal{V}$ is denoted by $exp = V(x; r_V) \rightleftarrows P(y; r_P)$ where x and y are the private values of V and P, respectively, and r_P and r_V are the *explicit* randomness that of the verifier and prover algorithm, respectively. In some protocols (e.g. password authentication) only explicit randomness is used. However protocols can also include the intrinsic randomness of $\mathcal{P}$ through user activities. The experiment can be extended to include an adversary $\mathcal{A}$ who interacts with the parties in the system. The expanded experiment is shown by $exp = (P(x; r_P) \rightleftarrows \mathcal{A}(r_A) \rightleftarrows V(y; r_V))$. A participant in a protocol instance has a *view* consisting of all its inputs, coins, and messages that it can see. The view of $\mathcal{A}$ includes all its communications with P and V. At the end of a protocol instance the verifier V outputs $out \in \{0, 1\}$ which is 1 if the authentication claim of the claiming prover is accepted, and 0 otherwise. The prover does not have an output. We use $\Pr_r[E : exp]$ to denote the probability of the event E in the protocol instance, and r to denote that random coins used in the protocol.

Definition 1. *A* Hard to Delegate (HtD) Authentication *system is a tuple (Reg, P, V) defined as follows. Reg is a registration protocol, run between $\mathcal{P}$ and $\mathcal{V}$ that takes a security parameter s, explicit randomness r and implicit randomness of $\mathcal{P}$, and outputs (s_P, s_{V_p}) (denoted by $(s_P, s_{V_p}) \leftarrow Reg(1^s, r, V_{reg}, P_{reg})$), where s_P and s_{V_p} are the values given to the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, respectively. We assume the protocol is always played honestly by the participants (secure registration) and treat it as a single function outputting the pair (s_P, s_{V_p}). The protocols satisfy the following properties.*

1. *Termination:*
   $(\forall s)(\forall r; r_V)$ if $(s_P, s_{V_p}) \leftarrow Reg(1^s; r, P_{reg}, V_{reg})$, and for any run of the protocol $(R \rightleftarrows V(s_P; r_V))$, between the verifier and an (unbounded) prover algorithm R, V halts in $Poly(s)$ computational steps;
2. *δ-**correctness**: $(\forall s)$ we have*

$$Pr\left[out = 0 : \begin{array}{c} (s_P, s_{V_p}) \leftarrow Reg(1^s; r, P_{reg}, V_{reg})) \\ P(s_P; r_P) \rightleftarrows V(s_{V_p}; r_V) \end{array}\right] \leq \delta$$

   where $s_P \leftarrow Reg(1^s, r, V_{reg}, P_{reg})$, $\Pr[out = 1 : exp]$ is the probability that verifier outputs 1 after the experiment is completed and the probability is over the randomness $\{r; r_P; r_V\}$.
3. *ϵ_d-**Delegation resistance** (ϵ_d-HtD)]*
   The probability that an adversary $\mathcal{A}$ (helper colluding with the user) successfully emulates $\mathcal{P}$, given access to registration information (s_P, s'_V), and after

[1] A vector of biometric feature points such as fingerprint minutiae that is collected from a user during an authentication session fits this definition also.

observing a number of instances of the authentication protocol $P \rightleftarrows V$, is bounded by ϵ_d:

$$Pr\left[out = 1 : \begin{array}{l} (s_P, s_{V_p}) \leftarrow Reg(r, P_{reg}, V_{reg}) \\ P(s_P) \rightleftarrows A_1 \rightleftarrows V(s_{V_p}) \\ A_2(s_P, s'_V, View(A_1), aux) \rightleftarrows V(s_{V_p}) \end{array}\right] \leq \epsilon_d$$

Here $(s'_P, s'_{V_p}) \leftarrow Reg(r, V_{reg}, P_{reg})$, is obtained by the interaction between $\mathcal{P}$ and a simulated verifier, and then given to $\mathcal{A}$ by $\mathcal{P}^*$. The adversary is shown by a pair of algorithms, (A_1, A_2). A_1 observes authentication sessions between $\mathcal{P}$ and $\mathcal{V}$, and provides its view to A_2. We use aux to denote other side information that $\mathcal{P}^*$ gives to $\mathcal{A}$.

Remark: Non-delegatability is an insider collusion attack. A corrupted registered participant $\mathcal{P}^*$ colludes with the helper $\mathcal{A}$, and gives them their registration information s_P as well as s'_{V_p} that is obtained by simulating the *Reg* protocol. Note that s'_P and s'_{V_p} will have the same distribution as the same intrinsic randomness of $\mathcal{P}$ is used. Security against delegation attack implies security against impersonation attack which is an outsider attack. This can be seen by using $A()$ with no privileged inputs instead of $A(s_P, s'_{V_p}, aux)$.

3 Authentication Games

To construct an HtD authentication system we use a challenge-response protocol where the verifier sends a challenge to the prover and receives a response. We use the following terminology and definitions. A *feature with respect to a game*, or a feature for simplicity, is a random variable that is associated with a game play and can be measured in each instance of the game play. An *identifying feature* is a complex function of one or more identifying personal traits. The measured value of a feature in a game instance is called a *feature point.* A *feature vector* is a vector of feature points that are collected in a game play.

3.1 An Authentication System Using Games

We consider the same setting of Section 2.1, and a two phase authentication system. A prover registers by participating in the registration protocol that is run by the verifier (or a trusted third party) and generates a *profile* of the user. In each instance of the game (a round of challenge-response), a vector of b feature points $F = (f_1, f_2, \ldots, f_b)$ is sampled, and sent to S as the response to the challenge. Let $R_P(n)$ denote a sequence of n feature vectors $F_1, F_2, \ldots, F_n$, that are collected in n consecutive runs of the game. $R_P(n)$ is the user *profile* held by $\mathcal{V}$. That is, *Reg* algorithm, here the n times game play, is used to produce the profile $R_P(n)$. (For example in our target shooting game, the user will have n runs to throw the arrow at the target.) Note that a user $\mathcal{P}$ can always simulate V algorithm and construct $R'_P(n)$. Assuming $\mathcal{P}$ game play is stationary, $R'_P(n)$ will have the same distribution as $R_P(n)$. The set of users' profiles forms the profile database DB, that will be used to verify users. During the authentication phase a user will be presented with n' consecutive challenges (game instance) one by one, and the collected set of n' responses (feature vectors) form $R_P(n')$ that will be used by the verification algorithm, to decide whether $R_P(n)$ and $R_P(n')$ are generated by the same $\mathcal{P}$ (same intrinsic distribution). Let $\mathsf{mtc}(R_{P'}(n'), \mathcal{P})$ be a matching algorithm that matches a given set of feature vector $R_{P'}(n')$, against the stored profile $R_P(n)$ of the user $\mathcal{P}$. The matching algorithm uses a distance function (Section 4.2) to compute the distance between $R_{P'}(n')$ and $R_P(n)$, for all $\mathcal{P} \in DB$ and outputs Accept (1) or Reject (0) if the distance was lower than a threshold.

Correctness and Security. For correctness, the distance between $R_P(n)$ and $R_P(n')$ must be small for the same user, and the distance between $R_P(n)$ and $R_{P'}(n)$ must be large for any two distinct users in DB. For security, $\mathcal{P}'$'s response must not result in the matching algorithm to output 1, assuming $\mathcal{P}'$ is given the simulated profile of $\mathcal{P}$ (i.e., $R'_P(n)$). We formalize these requirements as follows.

Definition 2. *A* $(b, m, n, (\alpha, \beta), \gamma, \mathsf{mtc})$-Authentication Game *is a game played between a user $\mathcal{P}$ who is a user in a set of m users, and the server S. In each instance of the game play a vector of b feature points, $(f_1, \ldots, f_b)$, is sampled. The user profile $R_P(n)$ consists of n feature vectors that are sampled in n consecutive rounds of the game. The matching algorithm* mtc *measures the distance of $R_P(n')$ to user profiles in DB, and outputs* $\mathtt{1}$ *if the distance is less than a threshold.*

1. (α, β)*- correctness:*
   - *α-FRR: For $n' < n$, the algorithm* mtc *outputs 1 with high probability given $R_P(n')$ and P:*

$$\Pr_P \left[\mathsf{mtc}(R_P(n'), P) \neq 1 \right] \leq \alpha.$$

   - *β-FAR: For a user $\mathcal{P}$, the probability that $R_{P'}(n')$ of user $\mathcal{P}' \neq \mathcal{P}$ is matched as $\mathcal{P}$ is bounded:*

$$\forall P, \Pr_{P' \neq P} \left[\mathsf{mtc}(R_{P'}(n'), P) = 1 \right] \leq \beta.$$

2. γ-Hard to Emulate (HtE)*: A game satisfies HtE if it is "hard" (measured empirically by the required time and training) for $\mathcal{A}$ to play in lieu of $\mathcal{P}$ and result in* mtc *to output $\mathcal{P}$ as the matched user. We assume $\mathcal{A}$ has $R_P(n)$, and additional information including possibility of observing game play of $\mathcal{P}$, denoted by Obs. Let $R_A(n')$ denote a set of n' feature vectors collected from $\mathcal{A}$'s game play when it is playing in lieu of $\mathcal{P}$. We require*

$$\Pr_{A \neq P} \left[\mathsf{mtc}(R_A(n'), P) = 1 \mid \mathcal{I} = \{R_P(n), Obs\} \right] \leq \gamma,$$

   holds for all P where $\mathcal{I}$ denotes the additional information available to $\mathcal{A}$.

Proposition 1. *An* $(b, m, n, (\alpha, \beta), \gamma, \mathsf{mtc})$*-authentication game is an authentication system satisfying definition 3, providing* δ*-correctness, and* ϵ_d*-resistance against delegation attack. We have* $\delta = \alpha + \beta$*,* $\epsilon_d = \gamma$ *and* $s = f(b, m, n)$*.*

The proposition follows by comparing definitions 1 and 2 and noting that in an authentication game, errors in honest game play of users will be in the form of FRR and FAR. A more detailed argument will be provided in the final version of the paper.

Feature Selection. Features are in general complex functions of multiple personal traits and can range from those that are mostly skill based and so learnable, to those that have deeper cognitive and behavioural base and so harder to learn by others.

Orthogonal Features. Our experiments show that one can use more features to increase accuracy of user authentication. In some cases a similar level of distinguishability can be obtained by reducing the number of features that are less correlated.

Tightly Coupled Features. For security against delegation attack features must be "hard" to transfer to others. For example, choosing objects in categories (e.g. clothing, pets) can be considered a personal trait that can form a feature in a game play. However such preferences cannot be used for HtE authentication as one can effectively pass on their preferences to others. To reduce the success chance of delegation, tightly coupled features can be chosen. These are dependent pairs and an attempt to change one will affect the other. For example, precision and speed of doing a task are tightly coupled features and increasing precision needs higher concentration and so more time, which will decrease the speed of performing the task. Tightly coupled features must be transferred together and this increases the difficulty of training the helper. In the above example training the helper to mimic higher precision of a skilled player should be together with mimicking their higher speed of playing the game.

3.2 Deploying Authentication Games

We analyzed the proposed authentication mechanism assuming a system design that enforces authentication by playing the game. *1- Overtaking network communication:* where the helper injects data packets directly into the network without playing the game. This attack would be successful if a fixed game is used and the user's response can be recorded. *2- Modifying the game client:* where the client software is modified to change the data input by the user to match the stored profile. *3- Automated game play (bot)* where a software is trained to emulate the behaviour of the legitimate prover in the game play. In our game, each challenge is freshly generated and developing a software agent that can learn the user behaviour in a complex game play requires major effort in learning theory and implementation to produce correct response in real time. In Appendix 6 we outline the prevention mechanisms against these attacks.

4 A Proof of Concept HtE Game

Our proposed HtE authentication game is an archery target shooting game. The game has a number of levels. In each level eight features, three primarily skill based, and five mostly behaviour based, are measured. More details on these features are given in Section 4.1. The game provides a clear goal for users to focus on. This is important for providing consistent game play statistics.

4.1 The Game Design

The implementation uses a 2D Physics engine to simulate the shooting of an arrow towards a target. The player drags and tilts the arrow (for example by using mouse) to choose the initial speed and the angle of throw, and release it to the target. The user wins if the arrow hits the centre of the target

Features Selection. In each shot of the arrow the following features are sampled. t_1-*Hit Error.* The distance between the arrow and center of the target after hitting. This is a floating point number in the interval $[-120, 120]$ as shown in Fig. 2.

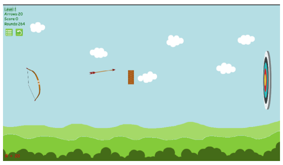


Fig. 1. Screen-shot of the game

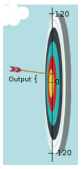


Fig. 2. Hit error

t_2-*Aiming time.* The time in milliseconds that it takes for the player to aim and shoot at the target. This is the time difference between the start of dragging and when the arrow is released which is a positive floating point number in $(0, 10]$. t_3-*Wait time.* The time in milliseconds that it takes for the player to begin dragging a new arrow, after the game is reset . This is a positive floating point number in the $(0, 5]$. t_4, t_5-*Relative initial Mouse click coordinates.* The x, y coordinates of the mouse initial clicking on the screen to drag the arrow, relative to the coordinate of the arrow's tail as the center. These are two floating point numbers greater than 0 and independent of the screen resolution. t_6, t_7-*Initial velocity and angle.* The velocity and relative angle of the arrow when it is released toward the target. t_8-*Miss count.* The number of misses between each two successful shots.

We note that the only varying parameter in the game that affects the measurements of features (e.g. t_1 and t_7) is the target location (in level 4). For both features, we measure them relative to the location of the target center. This makes the user feature points independent of the game parameters and our experiments shows stability of the values of these features over time. Features t_1, t_2, t_6, t_7 and t_8 are mostly based on personal traits such as concentration and cognition, reaction time and coordination: they measure precision and speed of a player in aiming at the target (selecting the angle and velocity of the arrow) considering the variable parameters of the game. Features t_3, t_4 and t_5, measure traits that are mostly subconscious including personal preference in where the arrow is grabbed. Our experiments show that decreasing the hit error and aiming time at the same time, is hard. Thus the pairs (t_1, t_2) and (t_2, t_8) are tightly coupled features: a player trying to decrease the hit error, needs to increase the time of aiming at the target. Our experiments suggest that the features $t_1, \ldots, t_8$ are stable and using consecutive measurements can identify users in a group (Section 5). Removing each of the features from the user game-play will reduce FAR and FRR.

Game Levels. The game design has evolved over a period of 2 months as we performed continued tests with 4 local participants. For our final evaluation using Amazon Mechanical Turks, we used 4 levels. Our observations on the affect of the design on the correctness and security are summarized in Section 5. In the first three levels of the 4 level game, the location of the target is fixed. The first level is the easiest: the target is fixed in the center of the screen and the player has to choose the speed and the angle of throw, and hit the center of the target. In the second level, there is a blocking wall that prevents the player to shoot at the target in straight line (Fig. 1. The player must adjust the angle and speed to prevent hitting the wall. The third level is the same as the first, but the target has a vertical periodic (sinusoidal) movement, and the player must predict the location of the target before releasing the arrow. The forth level is different from the previous 3 levels: the target will jump around and changes its location. It also fades away, and so forces the player to release the arrow within the time period that the target is visible. Otherwise the chance of hitting the target reduces.

4.2 Verification Function

The verification function is a matching algorithm that matches the user response in an authentication attempt against a stored profile. The stored profile $R_P(n)$ is a set of $n = 120$ feature vectors that are collected during the registration phase when the correct user is playing n rounds of the game challenge and response. Each authentication attempt consists of $n' = 30$ rounds of the game challenge and responses, where a user claims identity $\mathcal{P}$. The user profile is stored in the database DB indexed by the user identity and is used to match $R_P(n')$. We experimented with a number of candidate matching algorithms including SVM and random forest method, and chose the following algorithm because it provided the best accuracy (lowest FAR and FRR). The verification function takes as input two sets of feature vectors $R_P(n)$ and $R_{P'}(n')$ and outputs a bit, 1 or 0. The verification function estimates the probability distributions of features using the two sets of feature vectors, and compares the two distributions using statistical distance.

Converting Samples to Distribution. To construct a probability distribution for a feature from the profile (or an authentication attempt), one can construct the corresponding histograms (by defining bins and counting the number of samples in each bin), and then find a suitable parametrized density function that fits the data. Parameters of the density function will be determined using a goodness of fit algorithm, resulting in the probability distribution.

Our empirical results showed that *cumulative distribution function* (cdf) is more effective in distinguishing users. Our goal was thus to construct the cdf associated with a set of feature vectors, $R_P(n) = (F_1, F_2, \ldots, F_n)$, where $F_i = \{f_{1i}, f_{2i}, \ldots, f_{bi}\}, i \in [n]$. Here f_{ji} is the ith measurement of the feature f_j. Constructing the cdf of a multi-dimensional variable depends on the order that the variables are considered (corresponding to feature) and so the final distribution will depend on this order. To overcome this problem, we construct the cdf of each variable independently, use each to calculate a score for the corresponding feature in the authentication data, and then combine the results using the weighted average of these scores. To estimate the cdf of a feature f_j, we first extract the values of f_j from the set of feature vectors $C_j = \{f_{j1}, f_{j2}, \ldots, f_{jn}\}$. Assuming that the elements of C_j are samples of a distribution X, we want to estimate $\mathsf{cdf}(X)$ given by $\mathsf{cdf}(x) = \Pr[X \leq x]$, for a probability distribution $\Pr(X)$. Since we do not have the probability distribution X, we estimate the cdf which we call *empirical distribution function* (*edf*) by, $\mathsf{edf}_{C_j}(x) = \Pr_n[X \leq x] = \frac{1}{n}\sum_{i=1}^{n} I(C_{ji} \leq x)$, where C_{ji} is the ith element in C_j, and the function I returns 1 if the input condition is true and 0 otherwise. Thus $\mathsf{edf}_{C_j}(x)$ outputs the fraction of the sample points below value x.

The Distance Function. Given two sets of samples C_j, C'_j of size n and m respectively, we calculate the score as,

$$\mathsf{score}_j = \left(\frac{mn}{m+n}\right)^{1/2} \max_x \left|\mathsf{edf}_{C_j}(x) - \mathsf{edf}_{C'_j}(x)\right|.$$

score_j measures the distance between the two empirical distributions associated with the two sets of sample data. This function had been used in the *Kolmogorov-Smirnov (KS)* test as a measure of similarity between two datasets. The KS test measures the probability that two datasets are generated by the same distribution. The score is illustrated in Fig. 3 for 4 features measured in the game.

Finally, for the two sets of feature vectors $R_P(n)$ and $R_P(n')$, we define the score as a weighted sum of score_j for $j \in [b]$, $\mathsf{score} = \sum_{j=1}^{b} w_j \mathsf{score}_j$, where w_j is the weight of the feature j. The *score* can be considered as a measure of the likelihood that two sets of feature vectors are drawn from the same multivariate distribution. For a given profile $R_P(n)$ of user P and a response set $R_P(n')$, the verification function outputs 1 if the score is less than a threshold τ.

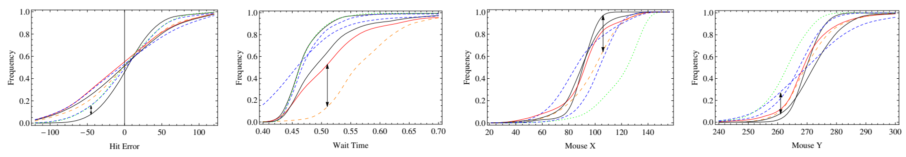


Fig. 3. The smooth edf of the features for 7 random users

5 Experiments

These experiments can be broadly divided into two groups and were performed over a six months period with a total of 186 users. There were 4 local users who participated in our experiments from the design stage, and allowed us to refine the design and parameter selection of the game. For the evaluation of our final design we used Amazon Mechanical Turks. The collected data from this latter group were filtered appropriately to exclude outliers as will be explained below. Our evaluation consists of two types of experiments, first for evaluating correctness and security as given in section 5.2, and second HtE property of the authentication game given in section 5.3.

Graphs in this Section. The figures used in this section illustrate the values of features measured through game play of users. The x-axis represent the feature value and the y-axis is frequency, or probability in the case of PDF or CDF. The graphs describe user behaviour as follows. Graphs for the timing of action such as targeting and wait time, shows the time spent for each feature. The user has spent less time for a feature, if the graph is towards the y-axis with higher peaks closer to value $x = 0$. For the feature "hit error", the user is more skilled in hitting the center of the target if the graph peak is around $x = 0$.

5.1 Considerations in Using Amazon Mechanical Turk

We had to ensure that users play the game consistently and to the best of their ability, and not at random and inconsistent way. To achieve this goal, users were instructed to play the game to achieve a minimum score at each level of the game. The minimum was set to be achievable by the weakest users. In each phase, the users were required to play the game for a required number of rounds without delay in between rounds. We measured timing parameters from the game to verify the users followed the requirements.

We note that feature measurements in general will be affected by the device and software platform including screen resolution or CPU speed. This is a known problem in behavioural authentication system that can be handled by considering multiple profiles for each user and introducing appropriate restrictions during deployment of the system.

5.2 Experiment 1: Correctness and Security

For correctness and security evaluation, we recruited approximately 150 Mechanical Turks, with only 97 of them passing our minimum requirements. Thus 101 users were used in the experiment, including our colleagues.

In the registration phase we collected 120 feature vectors (120 shots to the target) from each user, and in the authentication phase collected 30 feature vectors (on average taking around two minutes to complete) . The data for both phases were collected during a 6 hour period with roughly an hour in between registration and verification. This is to remove effect of learning, change of user experience and the like in measuring correctness. We will deal with these issues separately in Section 5.2.

Correctness (Single User). Our experiments showed that the measured features are fairly stable for user's recorded profiles. This means that the change in the values are so small that does not affect the matching algorithm. We examined users' data in two consecutive time slots and then constructed a histogram of the measurements. Fig. 4 is the histogram (`cdf`) for the two consecutive measurements for the two features, hit error and aim time, for one user.

In this experiment, we measured the stability of feature values during registration and authentication phases. We measured the distance (as described in Section 4.2) between the profile of the users constructed in the registration phase, and the measured feature vectors during authentication phase. The graphs in Fig. 4 are the histograms of the measured feature points of two features, hit error and aiming time. In each graph, the feature points during the registration and authentication are plotted separately. For 91 users (out of 101), the distance function outputs a very small difference between the registration and authentication data. This shows the stability of features during the two measurements indicating correctness of authentication game. Fig. 4 shows stability of measurements when performed in two consecutive time slots.

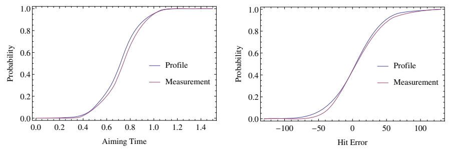


Fig. 4. User verification accuracy; measurements matching the profile

User Learnability and Profile Update. An important issue is the usage of the system over time. When a user profile is constructed at time t_1, one expects all (most) authentication attempts at times $t > t_1$ be successful. However, the change in the user's behaviour and skill over time could result in failed authentication attempt. We asked users to make login attempts over a period of 5 days. Each user on average made 20 login attempts at each level of the game. Fig. 5 shows the change in user behaviour over this period.

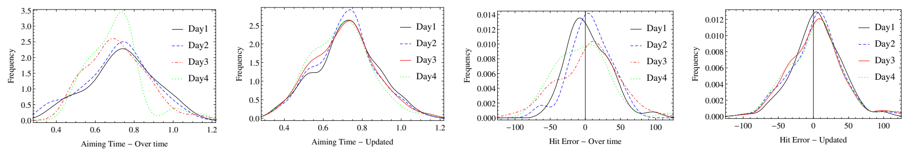


Fig. 5. The smooth histogram of the feature points for 1 user, illustrating how the features change over time.

There are 4 sub-figures in Fig. 5 illustrating the changes of two features, namely aiming time and hit error, over time. All sub-figures are extracted from one user data but the trends were the same for all users. The two sub-figures on the left show the behaviour of the user over time for 5 distinct measurements in the order numbered in the sub-figure legend. As shown in the sub-figures on the left, the behaviour and skill of users change over time and this can result in higher false negative in the matching algorithm. For example for the sub-figure related to aiming time feature, as the user becomes more experienced, less time is spent on aiming the arrow. For example, comparing the graphs on days 1 and 3, the peak of the graph 1 is on feature point 0.75 (seconds) compared to the peak 0.7 (seconds) in graph 3. The average value of aim time decreases as the user becomes more experienced in the game. For the sub-figure related to the hit error, the user's behaviour changes over time, but not necessarily towards lower error.

To compensate the affect of behaviour and skill change, the profiles of users were updated upon each successful login. The sub-figures on the right of Fig. 5 illustrates how updating the profile alleviates this problem. In the sub-figures on the right, the measurements are performed in the same order. For the measurements on days 1 and 3 the profile is updated, and authentication measurements on day 2 and 4 are compared against profile 1 and 3 respectively for verification. The results show that profile update is an important factor in accurate authentication over time. Without profile update around 70% of the authentication attempts (average over all users) failed, and this was mainly after a number of successful verification attempts. With profile update the same collected data showed 93% success rate in verification.

Security Against Impersonation: Multiple Users. Here the goal is to evaluate performance of the system in detecting a false claim: that is a user P' claiming to be P. In this experiment, we used the matching algorithm of Section 4.2 to evaluate how the feature points can distinguish users. The threshold was set to have a low FAR (level 4). Fig. 6 and 3 illustrate the histogram (pdf and cdf resp.) of feature points of 7 users' profiles. The user's histograms were distinguishable and the matching algorithm could correctly verify 91 out of 101 users. From the 9 users who were not verified, 4 were very close to the verification threshold. The other 5 users (all from Mechanical Turk) were far from their

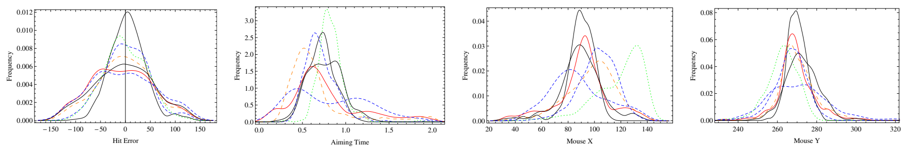


Fig. 6. The smooth histogram of the feature points for 7 users, illustrating how the features can distinguish among a group of people.

profile. However because of using Turks, it was not possible to ensure that the game plays were generated by the same user.

Entropy of the Authentication Information. The simplest attack on an authentication system is guessing attack where the attacker guesses the responses to the challenges. We used min-entropy which is the best success chance of guessing a variable, to measure guessability of a user profile. The measurements used NIST tests for estimating min-entropy explained in [BK12]. The measurements shows that the feature vector for each shot in the game has on average (over all users) at least 32 bits of entropy and so for 30 shots guessing entropy is $960 = 30 \times 32$ bits, making guessing attack impossible.

5.3 Experiment 2: HtE Property

We considered HtE property in the following scenario. A user registers to the system by playing the required number of game instances. The authentication information are passed on to a helper who will try to authenticate as the user. To evaluate HtE property, we considered an experiment where a group of users (helpers) all (independently) aim to emulate a target user. This would give us an estimate of the fraction of population who could successfully emulate the target user. Intuitively, this fraction would depend on the skillfulness of the target player. We chose two skill levels: a higher skill level and a lower skill level. There were a number of challenges in performing the experiment. Firstly, we had to ensure that the users (helpers) are incentivized to do their best to emulate the target users. We provided this incentive to Mechanical Turks who played the role of the helper, by offering a bonus of $20 for the task of successful emulation of the target, in addition to the standard payment. We also provided information that were "helpful" to the Turks so that they can modify their game play towards the target user. Providing plain user profile was soon proved to be not useful. Therefore, we initially provided a set of information about the target user to the Turks, and then provided feedback after each authentication attempt. The set of information included i) a video recording of the target user playing the game, and ii) the statistics of the feature points in the target user's profile such as maximum, minimum and average values of each of the features. The feedback information included i) the statistics of the feature points in the Turk's data from each verification attempt (Table 2), ii) direct instructions on how the user should change behaviour to match the target user as in Table 1, and iii) the graphs comparing the distribution of the target user's profile and the Turk's verification attempt as in Figures 7 (c,d).

In our experiments we hired 2 local users, one a more skilled user and a less skilled one, as delegation targets. We hired 15 Mechanical Turks and 2 local users to emulate each of the target users. The 2 target users also tried to emulate each other, so the total number of participants (who satisfied our requirements) was 36. The Mechanical Turks were selected with varying skill levels, based on their previous scores, and new users who had not played the game before. These choices were to make the experiments unbiased. The user tried to emulate the behaviour of the target users at least 20 times over a period of 5 hours. The task was allowed to be continued if the users were interested in making more attempts to win the bonus payment. In total we had, 904 and 1606 login attempts to emulate the behaviours of the user 1 and 2 respectively.

Our first observation in evaluating HtE property is that any false positives in authentication phase implies that HtE property will not be satisfied. In other words, if the authentication algorithm matches authentication attempts of user A to the profile of user B, this implies that user A can emulate the behaviour of user B. So in experiment 5.2, we counted the number of users who could authenticate as another user. Note that in experiment 5.2, the users were not asked to emulate the behaviour of another user. But their data was close enough to another user that resulted in a false positive.

HtE Property of the Game. A player X from Mechanical Turk was given the following information about the target player Y (local): the record of feature measurements, feature statistics (average, min, max), graphs of feature points (as used by the verification algorithm), the information from visually observing the game play of Y and instructions on how to change behaviour to get closer to user Y. With this, player X had to emulate user Y in several authentication rounds, each consisting of 30 game plays. After each round, we provided feedback (increase or decrease the feature values) to X on how to change their game play to get closer to the target. The player X was also told about how the matching algorithm rated the feature points compared to Y's profile. We had asked player X to play as themselves in their first attempt so that we could compare and measure the progress in the behaviour emulation. We repeated this experiment with direct supervision of the 4 local users, trying to emulate the behaviour of the two target users.

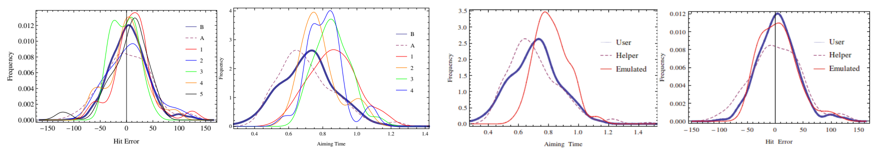
Table 1. Instructions provided after each attempt

Increase aiming time by 0.5 seconds.
Decrease wait time by 0.3 seconds.
Decrease hit error by 10 pixels.
Increase Mouse X by 20 pixels.

Table 2. Statistical information of behaviour

Feature	Min	Max	Average
Aiming time	0.5	2.3	1.2
Wait time	0.2	1.3	0.7
Hit error	-89.45	56.31	5.3

The Affect of Tightly Coupled Features. From the 36 users attempting to emulate the behaviour of our players, three Turks could emulate the behaviour of the local users. However, only one Turk could repeat their success in emulating the behaviour of the weaker user such that the matching algorithm outputs 1 in around 26% of the attempts. We note that the behaviour of this participant was relatively close to the weaker user in their first attempt. The other two participants could only emulate the behaviour of the local users once or twice in all their attempts. In total, for 1606 attempts to emulate the behaviour of the stronger user, only 2 attempts were successful and matching algorithm output 1 with 95% success. For the weaker local user however, out of 904 emulation attempts, 13 attempts were successful.



(a) (b) (c) (d)

Fig. 7. (a,b) HtE property; measurements for two single features (c,d) Increase in hit error results in increase in aiming time.

Fig. 7 (a,b) illustrates the attempts made by a user to emulate a second user. The histogram of feature measurements of user X is shown in dashed line before passing on Y's information. The histogram of feature measurements of user Y (from Y's profile) are shown in a blue thick lines. The remaining graphs correspond to attempts made by user X to emulate the behaviour of user Y for two sample features, hit error and aiming time. As shown in the figures, user X has lower hit error initially, while aiming time is roughly similar to Y. But an attempt to increase the hit error results in longer aiming time, even when X is trained to emulate the behaviour of user Y. Therefore user X could not emulate both features at the same time. This is illustrated in Fig. 7 (c,d) for one attempt to simulate the behaviour of another user. In general, time and coordinate related features were harder to emulate. For example the difference in wait time of two users, although it could distinguish the users, but was not significant so that a user can emulate the exact delay of the second user[2].

Stronger Versus Weaker Users: Our experiments showed that it is easier to emulate the behaviour of weaker users compared to the more skilled users and for the former group, a helper could improve its emulation of the target user. For strong users however, some of the users could not have any progress in their emulation attempt and the rest could not get close enough to the behaviour of the target user.

[2] The user trying to emulate a second user had this comment: "How can I delay for 0.3 seconds more in each game play?!"

Game Design and Parameters. The game was developed over a period of few months taking into account the effect of varying parts of the game on accuracy of authentication and the HtE property. Using the feedback of the 4 local users, the game variables including the gravitational force, the speed of target movement, appearance of obstacle in the arrow path, and making the target hidden, were modified to examine their effect on correctness, security and HtE properties. We finally selected 4 levels (described in Section 4.1) for the main experiments. Variations such as target movement in the game can significantly reduce convergence of user profiles. This observation was supported by our experiments as shown below. The value of FRR for the 4 levels of the game is 28%, 18%, 24%, 9% and value of FAR is 12%, 6%, 13%, 6% respectively. As can be seen, level 4 results in the lowest FRR and FAR and thus is more suitable for providing non-delegatability. The target in level 4 fades in and out in different locations and this makes it harder to achieve higher scores. The issue that may rise here is that the variations may cause instable feature measurements over time. However, relative measurements (to the variations in the game) can mitigate this issue.

6 Deploying Authentication Games

In the following, we discuss possible attacks and prevention mechanisms on authentication games. There is an ongoing research on the topic of cheat prevention in online games that enables hackers to modify the client, or change the network communication so that they win without playing. A survey and classification of these attacks can be found in [WK12]. The success of an online multi-player game is very much dependent on its fairness among players and thus gaming industry invests on developing anti-cheating mechanisms due to its financial significance.

In the following sections, we will summarize the methods in this line of work that can be used to protect a HtE game against the three mentioned attacks.

Tampering with Network Communication. In this attack the delegatee uses a trained software that can emulate the behaviour of a legitimate prover, to bypass the game client and sends the information to the verifier over network. To prevent this attack we assume a secure communication between the verifier and game client. This can be achieved by obfuscating a shared key K inside the game client. We assume this key is not retrievable/modifiable by the users of the system, neither the prover, not the delegatee. Note that *we do not restrict access to the same game client software by any party*, so the delegatee may acquire a copy of the game client with the same shared key. Assuming the shared key, a secure authentication mechanism can be implemented in the game client to prevent any tampering with the network, including replay attack where the delegatee only replays the responses from the prover. We note that this is not a full proof solution, but it is assumed in many cheat-proofing mechanisms for games [HARD10] as it effectively prevents cheating.

Game Client Modifications. The delegatee might modify the client to bypass the authentication system in two ways. First by installing a cheat along with the game client as a patch or loadable module to help in emulating the behaviour of the prover, and in second method by retrieving/modifying the shared key with the verifier to be able to tamper with the network communication. To mitigate these attacks, authors in [TBB12] propose to symbolically execute the client to find the constraints on the state of the client implied by the responses received from it, and then using constraint solvers to find if such constraints could be generated by user input. An extension of this approach was proposed in [HARD10] which uses Accountable virtual machines (AVM). In this approach, the game is run in a virtual machine that monitors the state of the game during user game play and outputs a log of the game events (e.g. mouse click, key stroke, etc) which will be sent to the verifier. Having all the logs, the verifier can simulate running the game with the events in the log to find inconsistencies. There are also solutions based on tamper-resistant hardware [BM07] that use a dedicated hardware to check the state of the client.

Automated Game Play (bot). A game bot is a software/hardware agent that can emulate game play. In this attack, a game bot can be trained to be able to emulate the behaviour of the prover, without client modification or tampering with the network. For example one type of game bots can generate the sequence of mouse clicks and key strokes to play the game, by image processing the game environment. Depending on the graphics of the game, such tools can get very complex and harder to implement. There are general protection mechanisms to mitigate these attack such as Intel hardware protection mechanism [SGJ07], and software techniques such as human interactive proofs (e.g. Captcha) [MY12]. In [GWXW09], human observational proofs (HOP) are used to distinguish between human and bots. HOPs differentiate bots from human players by monitoring actions taken by the player that are difficult for a bot to perform. [CPC08] tries to distinguish human behaviour from bots by arguing that certain human behaviours are difficult to perform by a bot because they are AI-hard. Note that the methods in [GWXW09, CPC08] collect feature from game play and can be simply incorporated into our proposal by unifying the collected features and doing further analysis on the feature vector to detect bots, and then verify the identity of prover.

D-MiM Attacks. In a delegation Man-in-the-Middle attack (D-MiM) the helper forwards the challenge to the colluder, receive its response and passes it on to the verifier. This attack is only possible if the colluding prover is online at the time of the challenge. Although this is a valid attack if the time is coordinated before hand, it becomes increasingly hard if the verifier use the system in *continuous authentication* mode (e.g. in scenario of work-at-home) and send challenge blocks at random times to the user. Similar to other MiM attack, providing protection against D-MiM can be achieved by using extra mechanisms such as distance bounding protocols to verify the distance of the user from the verifier. Note that although distance bounding protocols are primarily for wireless environments, there are distance bounding protocols that work over the Internet. Distance bounding over wired networks has been considered in a number of works. Drimer et al. [DM07] proposed to use DB over wired networks to prevent relay attacks between bank terminals and smart cards. Watson et al. [WSNA+12] also proposed DB to estimate the location of a server over a wired network which describes a method to achieve an estimation error of 67 km for distance.

7 Concluding Remarks

We proposed a novel approach to challenge response authentication using behavioural biometrics in a complex activity. Exploring possible activities that can be used for user authentication, and feature selection for these activities is an interesting direction for future work. Another important direction for future work is privacy of user data. A user profile is a set of feature vectors that is only meaningful with respect to the activity. Developing a privacy model and evaluating it experimentally is also an interesting direction for future research.

Acknowledgement. This research is in part supported by Natural Sciences and Engineering Research Council of Canada.

References

[AFN13] Alayed, H., Frangoudes, F., Neuman, C.: Behavioral-based cheating detection in online first person shooters using machine learning techniques. In: 2013 IEEE Conference on Computational Intelligence in Games (CIG), pp. 1–8, August 2013

[BK12] Barker, E., Kelsey, J.: Recommendation for the entropy sources used for random bit generation, August 2012. http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf

[BM07] Balfe, S., Mohammed, A.: Final fantasy – securing on-line gaming with trusted computing. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 123–134. Springer, Heidelberg (2007)

[BSR+12] Bojinov, H., Sanchez, D., Reber, P., Boneh, D., Lincoln, P.: Neuroscience meets cryptography: designing crypto primitives secure against rubber hose attacks. In: Proceedings of the 21st USENIX Conference on Security Symposium, Security 2012, pp. 33–33. USENIX Association, Berkeley (2012)

[Car93] Carroll, J.B.: Human Cognitive Abilities. Cambridge University Press (1993)

[Cat57] Cattell, R.: Personality and motivation structure and measurement (1957). http://psychology.about.com/od/trait-theories-personality/a/16-personality-factors.htm

[CH07] Chen, K.-T., Hong, L.-W.: User identification based on game-play activity patterns. In: Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games, NetGames 2007, pp. 7–12. ACM, New York (2007)

[CPC08] Chen, K.-T., Kenneth Pao, H.-K., Chang, H.-C.: Game bot identification based on manifold learning. In: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, NetGames 2008, pp. 21–26. ACM, New York (2008)

[DBvDJ11] Denning, T., Bowers, K., van Dijk, M., Juels, A.: Exploring implicit memory for painless password recovery. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2011, pp. 2615–2618. ACM, New York (2011)

[DM07] Drimer, S., Murdoch, S.J.: Keep your enemies close: distance bounding against smartcard relay attacks. In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, SS 2007, pp. 7: 1–7: 16. USENIX Association, Berkeley (2007)

[Fid13] Fiddy, H.O.: Method and system for defeat of replay attacks against biometric authentication systems, US Patent 8,508,338 (2013)

[GWXW09] Gianvecchio, S., Zhenyu, W., Xie, M., Wang, H.: Battle of botcraft: fighting bots in online games with human observational proofs. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 256–268. ACM, New York (2009)

[HARD10] Haeberlen, A., Aditya, P., Rodrigues, R., Druschel, P.: Accountable virtual machines. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI 2010, pp. 1–16. USENIX Association, Berkeley (2010)

[MR00] Monrose, F., Rubin, A.D.: Keystroke dynamics as a biometric for authentication. Future Generation Computer Systems **16**(4), 351–359 (2000)

[MY12] McDaniel, R., Yampolskiy, R.V.: Development of embedded captcha elements for bot prevention in fischer random chess. Int. J. Comput. Games Technol., p. 2:2 (2012)

[PB04] Pusara, M., Brodley, C.E.: User re-authentication via mouse movements. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 1–8. ACM (2004)

[Rev08] Revett, K., Biometrics, B.: A Remote Access Approach. John Wiley & Sons Ltd. (2008)

[SGJ07] Schluessler, T., Goglin, S., Johnson, E.: Is a bot at the controls?: detecting input data attacks. In: Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games, NetGames 2007, pp. 1–6. ACM, New York (2007)

[TBB12] Tian, H.Y., Brooke, P.J., Bosser, A.-G.: Behaviour-based cheat detection in multiplayer games with event-B. In: Derrick, J., Gnesi, S., Latella, D., Treharne, H. (eds.) IFM 2012. LNCS, vol. 7321, pp. 206–220. Springer, Heidelberg (2012)

[TH13] Thanh Ha, T.: Us developer outsourced his job to china (2013). http://www.theglobeandmail.com/technology/how-a-model-employee-got-away-with-outsourcing-his-software-job-to-china/article7409256/3/

[WK12] Woo, J., Kim, H.K.: Survey and research direction on online game security. In: Proceedings of the Workshop at SIGGRAPH Asia, WASA 2012, pp. 19–25. ACM, New York (2012)

[Wor13] Wortham, J.: No tv? no subscription? no problem (2013). http://www.nytimes.com/2013/04/07/business/streaming-sites-and-the-rise-of-shared-accounts.html

[WSNA+12] Watson, G.J., Safavi-Naini, R., Alimomeni, M., Locasto, M.E., Narayan, S.: Lost: location based storage. In: Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop, CCSW 2012, pp. 59–70. ACM, New York (2012)

[YG09] Yampolskiy, R.V., Govindaraju, V.: Strategy based behavioural biometrics: a novel approach to automated identification. Int. J. Comput. Appl. Technol. **35**(1), 29–41 (2009)