IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Towards incorporating honeywords in n-session recording attack resilient unaided authentication services

ISSN 1751-8709
Received on 7th November 2017
Revised 23rd March 2018
Accepted on 4th May 2018
E-First on 18th July 2018
doi: 10.1049/iet-ifs.2017.0538
www.ietdl.org

Nilesh Chakraborty[1] ✉, *Samrat Mondal*[1]

[1]*Department of Computer Science & Engineering, Indian Institute of Technology Patna, Patna, Bihar 801103, India*
✉ *E-mail: nilesh.pcs13@iitp.ac.in*

Abstract: Unaided authentication services provide the flexibility to login without being dependent on any external hardware. n-Session recording attack resilient unaided authentication services (n-SRRUASs) are known for setting high security standards against different client side threats. However, because of their authentication procedure, the authors have identified that these services cope poorly with handling the server side issues. Though modern days' research heavily depends on the *honeywords* (or fake passwords) as a countermeasure of server side threats, they have shown that the *honeywords* cannot be directly applied to n-SRRUAS. The authors' analysis shows that the idea of incorporating the *honeywords* directly into an n-SRRUAS is particularly difficult as it prevents the system from storing passwords after applying password-based key derivation function or in the form of a hashed string. In this study, they have proposed few generic principles for incorporating the *honeywords* into n-SRRUAS and show that the proposed principles are sufficient for incorporating the *honeywords* into any n-SRRUAS. Furthermore, with the help of an existing n-SRRUAS, they have shown that the proposed idea is truly implementable in practice to fill the existing gap.

1 Introduction

Despite constant efforts for replacing the passwords since last two decades, password based authentication still dominates the modern forms of identity verification. Because of its tremendous popularity, passwords remain as the prime target for the attackers to threat the users' community at a large scale [1, 2]. Various attack models have been developed till date for compromising the users' passwords from both the client and server ends [3–5]. Recording attack [6, 7], responsible for the security breach at the client side, is one amongst the oldest threats on password-based authentication systems that reveals the many users' identity to the attackers.

1.1 Threat model

The literature in this domain suggests that *powerful passive adversaries* are responsible for performing the recording attack [6, 8, 9]. *Powerful* represents the fact that the adversaries can record, monitor, intercept and analyse each part of an authentication process (i.e. all inputs and outputs of the authentication procedure), except the pre-shared secret between the user and system. *Passive*, in contrast, describes the limitations of the adversaries as they cannot disrupt the authentication process or alter or create new content transferred during the login. The adversary may use any recording device (e.g. conceal camera) for performing this attack. Conducting this attack is quite feasible since small high quality cameras, even wearable ones, are already available at a low cost. The goal of this attack is to subsequently impersonate the genuine user from the recording evidence.

1.2 Defence strategy against the recording attack

There are following three components which build the security against this attack:

- user,
- the link between the user and system,
- system (or server).

For avoiding the recording attack, the role of each of these components is elaborated at next.

User's activity: A user at first receives a challenge ($\mathscr{C}$) from the system. Based on the original password ($\mathscr{P}$), the user then derives a response ($\mathscr{R}$) with the help of $\mathscr{C}$. Therefore, for a function f which maps $\mathscr{P}$ and $\mathscr{C}$ to $\mathscr{R}$, the computation from the user's end can be described in the form of the following equation:

$$f: \mathscr{P} \times \mathscr{C} \rightarrow \mathscr{R} \tag{1}$$

The system varies $\mathscr{C}$ in each authentication session and because of this, $\mathscr{R}$ also changes. In this way, indirect key entry (i.e. submitting $\mathscr{R}$ instead of $\mathscr{P}$) restricts the adversaries from getting the original password of the user. Based on the user's activity, below, we note the outcomes of the above discussion:

- *Note 1:* Indirect key entry refrains an adversary (performing the recording attack) from getting the original $\mathscr{P}$ of a user.
- *Note 2:* Like other authentication systems, we assume no involvement of the physical occlusion during submission of $\mathscr{R}$ by the users. Hence recording attack reveals $\mathscr{R}$ to the adversary.
- *Note 3:* As stolen $\mathscr{R}$ by the adversary changes in each authentication session, therefore, activity from the user's end successfully resists the phishing attack (occurs when a user is tricked into providing the correct answer to a malicious website).
- *Note 4:* Because of the varying $\mathscr{R}$, this kind of login setup is also capable of defending the key-logger based attack, capturing user's login credentials from the inputs of a keyboard.

Link's responsibility: A link establishes a bridge between the user and system and it exchanges the information between both these parties. The system uses the link for sending $\mathscr{C}$ to the user. After deriving $\mathscr{R}$, the user sends it back to the system via this link. It is important to note that a link can be categorised as follows:

- *Secure link:* It is assumed that a secure link never leaks any information to any adversary and hence, $\mathscr{C}$, sending via the secure link, never gets disclosed to the adversary [10].
- *Insecure link:* The information exchanged via this link can be intercepted by the adversary.

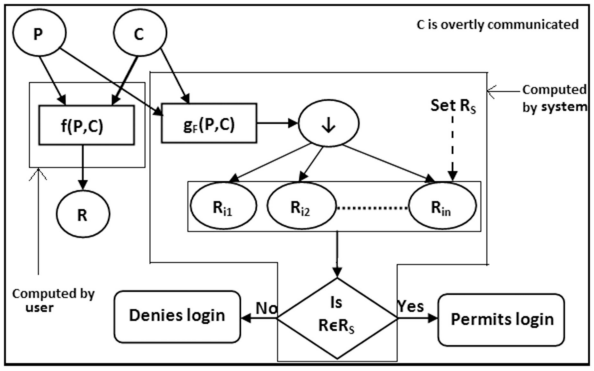


Fig. 1 *Computations carried out by both the user and system during the authentication. '↓' inside the circle implies a mapping operation. R_{ik} indicates k*th (≥ 1) *element of the response set R_S, generated by $g_F(\mathscr{P}, \mathscr{C})$ in the i*th *round of an authentication session*

However, many types of research in this domain conclude that the assumption behind the secure link is far from practical, and thus, the methodologies, working under this speculation, have very little significance in the real world [11]. Therefore, in this study, we always assume that the link between the user and system is insecure. An insecure link always discloses $\mathscr{C}$ to the adversary to fit into the considered threat model, discussed in Section 1.1.

- *Note 5:* An insecure link discloses $\mathscr{C}$ to the adversaries.

Before going into any further discussion, it is important to mention that an authentication service defending recording attack based on an insecure link can protect the $\mathscr{P}$ for n ($1 < n \leq \infty$) sessions before the adversary recover it ($\mathscr{P}$) from the recording evidence. In this literature, we have identified such authentication services as *n-session recording attack resilient authentication services* or n-SRRUASs. In the literature, there are many examples of n-SRRUAS. Some of the commonly known schemes can be found in [6, 7, 9, 12–16] and so on.

Computation at the server side: This part of our discussion is very crucial to get a glimpse of why the *honeyword*-based authentication technique (HBAT) [17] cannot directly be applied to an n-SRRUAS. Notably, to check the correctness of the submitted $\mathscr{R}$, the system first computes a set of probable responses. This set is created with the help of the stored $\mathscr{P}$ in the password file and the communicated $\mathscr{C}$ to the user. We denote this set of probable responses as R_S. Depending on the authentication service, R_S may be a singleton also [14]. Therefore, for a function g_F, deriving R_S from $\mathscr{P}$ and $\mathscr{C}$, the computation from the server's end can be presented in the form of the following equation:

$$g_F: \mathscr{P} \times \mathscr{C} \rightarrow R_S \tag{2}$$

The system only validates the user, if received $\mathscr{R}$ belongs to the set R_S. It can be noted that for a response window, R_W, containing all possible responses, R_S is a proper subset of R_W. The aforementioned discussion leads to the following conclusive remarks:

- *Note 6:* $\mathscr{P}$ in the password file must be stored in the plaintext format for forming the set R_S.
- *Note 7:* An n-SRRUAS needs a reference to the plaintext passwords for validating $\mathscr{R}$.

Crucial discussion: From the presented text so far, it may seem that an n-SRRUAS can avoid storing $\mathscr{P}$ in the plaintext format by doing the following adjustments. Equation (1) can be modified as

$$f: H(\mathscr{P}) \times \mathscr{C} \rightarrow \overline{\mathscr{R}} \tag{3}$$

where the transformation from $\mathscr{P}$ to $H(\mathscr{P})$ changes $\mathscr{R}$ to $\overline{\mathscr{R}}$. Also, (2) needs to be adjusted to

$$g_F: H(\mathscr{P}) \times \mathscr{C} \rightarrow \overline{R_S} \tag{4}$$

where the change from $\mathscr{P}$ to $H(\mathscr{P})$ transforms R_S to $\overline{R_S}$.

Following the above scenario, the system will authenticate a user only when it finds that $\overline{\mathscr{R}}$ belongs to $\overline{R_S}$. However, this may threat the usability standard and the working principle of an n-SRRUAS because of the following major concerns:

- *Concern 1:* Function f in (1) is executed by the human users. Therefore, if (1) is transformed to (3), then prior to executing f()
  - The users need to compute $H(\mathscr{P})$ from $\mathscr{P}$. However, manually deriving $H(\mathscr{P})$ from $\mathscr{P}$ is a complex procedure. Thus the processing of standard cryptographic hash function (e.g. MD5, SHA1 etc.), falls beyond the capacity of the human mind [18].
  - The user may use any computing device which takes $\mathscr{P}$ as input and outputs $H(\mathscr{P})$. However, while giving input to the computing device, $\mathscr{P}$ may be compromised under the recording attack. Also, usage of any external hardware for computing $H(\mathscr{P})$ violates the basic working principle of an n-SRRUAS, as the authentication system would not remain unaided any more.
- *Concern 2:* During registration, even if the system generates the hashed string corresponding to the users' chosen $\mathscr{P}$, the users should be incapable of remembering those because of their excessive length [19]. For example, if $\mathscr{P}$ be '*A1B3*', then the corresponding hashed string (based on the MD5 hashing technique) will be '*a8956926ac7ce89c37d5687368b37ac1*' (http://www.sha1-online.com/).
- *Concern 3:* Many n-SRRUASs accept a part of $\mathscr{P}$ as a parameter of g_F function (see detail in Section 5.1). For example, in the S3PAS authentication scheme [12], the password '*A1B3*', is decomposed into '*A1B*' in the first round, '*1B3*' in the second round, '*B3A*' in the third round and '*3A1*' in the last round. Also, like in [6], for a password of length ℓ, an n-SRRUAS may randomly select a substring of $\mathscr{P}$ from the index positions i to j ($1 \leq i, j \leq \ell$ and $i \leq j$) for validating a user. For a $\mathscr{P}$, stored in the hashed format, this kind of decomposition may not be possible.
- *Concern 4:* The graphical passwords, often used in defending the recording attack, cannot be converted into hashed format [9].

Because of the aforementioned issues, it is always preferable for an n-SRRUAS to maintain $\mathscr{P}$ in the plaintext format.

- *Note 8:* For protecting the original $\mathscr{P}$, if an n-SRRUAS maintains *honeywords* (or false passwords) in the password file, then it may create a confusion during identifying the original $\mathscr{P}$ from its corresponding R_S (see detail in Section 4).
- *Note 9:* In the absence of the *honeywords*, if an adversary gets an access to the password file of an n-SRRUAS, then all the original passwords can very easily be obtained from that password file and, this will completely break the security of the system.

In Fig. 1, we have illustrated the detailed responsibilities carried out by both the system and user during the authentication period for defending the recording attack.

We believe that all the notes can easily be understood from the presented text so far. However, for understanding the idea behind *Note 8, it may require some basic knowledge about the working principle of HBAT*. This is discussed in Section 3. Based on the aforementioned notes, next, we have built the motivation behind this work.

1.3 Motivation and contribution

Notes 1, 3 and 4 infer that an n-SRRUAS provides robust security against different client side threats. Notes 2 and 5 suggest that both $\mathscr{C}$ and $\mathscr{R}$ are overtly being exchanged between the user and system, and hence, without dependency on any external hardware, a user can prove her identity to the system. Being impressive so far,

Notes 6–9 indicate that these defence mechanisms cope poorly in handling the server side issues. Recent evidence shows that the password leakage from the compromised servers no more exists in theory only. For example, in 2013, almost 50 million passwords of Evernote were stolen due to the breach on the server side [20]. Giant web-based organisations like LinkedIn, Yahoo, RockYou have gone through the same misery [21]. This implicates that there is an absolute need for protecting the password against the server side attack. Unfortunately, in its current form, any n-SRRUAS ends up being vulnerable to the server side threat. To make it worse further, Note 8 strongly recommends that HBAT, saving passwords on many occasions [22, 23], cannot be directly applied to any existing n-SRRUAS, restricting passwords to be maintained in hashed format or after applying password-based key derivation function (PBKDF) on it. Mainly motivated by the aforementioned facts, we gist our major findings below:

- *Contribution 1:* We have provided a detailed analysis to show that migration to HBAT is not immediate for any n-SRRUAS.
- *Contribution 2:* We have proposed a few generic principles for incorporating the *honeywords* to an n-SRRUAS.
- *Contribution 3:* We have shown that against the server side attack, generated from the compromised password files, the proposed principles increase the security of n-SRRUAS from 0 to $((k-1)/k)\times 100\%$; where $k-1$ denotes the number of *honeywords*.
- *Contribution 4:* To show that the proposed concept is deployable in practice, we have applied our principles for adding the *honeywords* to the password files of an existing n-SRRUAS, S3PAS [12].

2 Related work

To the best of our knowledge, in 1991, Matsumoto and Imai's approach first gave a direction for handling the recording attack without being dependent on any external hardware [8]. Lately, in 2001, Hopper and Blum's protocol captured different aspects of n-SRRUAS in achieving the desired security goal [7]. Notably, the work by Wiedenbeck *et al.* [9], namely convex-hull-click, created a benchmark in 2006 in the domain of n-SRRUAS. The basic principle of convex-hull-click was extended by Zhao and Li [12] and Wu *et al.* [15] in 2007 and 2014, respectively. In 2008, Takada [24] came with another approach, FakePointer, to deal with this attack model. Being *n*-session resilient to the considered threat model, proposed ColorRings by Kim *et al.* [25] and ColorPIN by De-Luca *et al.* [13] showed the significant progress from the usability point of view. Though Ashghar *et al.*'s work in 2010 significantly improved the security standard against the recording attack; it failed to provide the desired usability standard. Recently, GOTPass by Alsaiari *et al.* [26], and PassMatrix by Sun *et al.* [27] have made their mark for building security against this attack scenario.

Along with the prevention, modern research is shifting towards detecting the attackers' activity. The detection technique is gradually gaining popularity under the light of HBAT [28]. Briefly speaking, this strategy obfuscates the attackers among several alternatives and, among them, only one represents the original password. With the knowledge of the actual password, though a user always selects the correct alternative, choosing a wrong option by an attacker detects the threat. Therefore, for k such alternatives, the probability of threat detection yields to $(k-1)/k$. To the best of our knowledge, Deception Toll KIt (DTK) (http://www.all.net/dtk) – developed by Fred Cohen in 1997, was one of the first publicly available tools that use deception in computer security. Following this, the deception technique has been used in many computer related domains. Kim and Spafford [29] showed that the *honey-files* (named as *planted files* in their contribution) can effectively guard the early version of *Tripware*. *Honeywords* have also been incorporated for encryption purpose and to detect threat on the password file [17, 30]. The concept has shown its potential in handling different network related issues too [23, 31]. Recently, this deception strategy has created its mark in smartcard based two-factor authentication techniques [32]. Above discussion suggests that the concept of *honeyword* enriches many security related domains. However, the idea has never been deployed in protecting the password files of any n-SRRUAS to deal with the server side issues.

3 Preliminaries

This section deals with some basic information which will help to understand the proposed concept, described in Section 5. We aim to cover the following three major topics under the scope of this section:

- attack principle on n-SRRUAS,
- HBAT password management policy,
- assumptions.

3.1 Attack principle on n-SRRUAS

As mentioned earlier in Section 1.2, from the recording evidence of an authentication session, a recording attacker obtains both $\mathscr{R}$ and $\mathscr{C}$. From this information, the attacker forms a set of probable secrets, which we denote as $\mathbb{S}_i$ (where i denotes the recording evidence of the ith authentication session). This activity from the attacker can be expressed in the form of the following equation:

$$g_{\mathrm{A}}: \mathscr{R}\times\mathscr{C}\rightarrow\mathbb{S}_i \tag{5}$$

where $|\mathbb{S}_i|>1$ and $\mathscr{P}\in\mathbb{S}_i$.

Understandably, both $\mathscr{R}$ and $\mathscr{C}$ change in each ith session and as a consequence, $\mathbb{S}_i$ too gets varied. The term $\mathbb{S}_i$ denotes the leaked information (which contains probable candidates for $\mathscr{P}$) from the captured evidence of the ith session. Leaked information from each session helps an adversary in reducing the size of probable candidate elements and thus, $\mathscr{P}$ eventually gets disclosed. For example, after recording n authentication sessions (where $1<n\leq\infty$) if intersection among these $\mathbb{S}_i$ reveals $\mathscr{P}$ to the adversary, then this phenomenon can be explained in terms of the following equation:

$$\mathbb{S}_1\cap\mathbb{S}_2\cap\cdots\cap\mathbb{S}_n=\mathscr{P} \tag{6}$$

The aforementioned attack is known as brute force attack on n-SRRUAS and this value of n varies based on an n-SRRUAS. In [11], the authors show that along with the brute force attack, different statistical attack can also diminish this value of n. In this context, it is important to note that though the value of n can be infinitely large in theory, it rarely passes a finite limit in practice without affecting the usability standard significantly [7] and thus, most of the methods in this direction restrict the value of n to a finite limit [6, 9, 12–16].

3.2 HBAT password management policy

The passwords chosen by the users' are found to be highly skewed in nature [33]. Because of their predictability, modern password cracking algorithms can invert almost all the passwords even if they are stored in the hashed format [5, 34]. To make the situation worse further, in Password'12, Jeremi Gosney showed that 12 h is sufficient to invert any password of length 6 [35]. To address this kind of threat, Juels and Rivest [17] introduced the concept of fake passwords (or *honeywords*), which can be stored in the password file along with the original password of the users. The purpose behind their proposal was to both detect and prevent the server side threat. As proposed in [17], a system, governed by HBAT, stores each user's login credentials in the following manner:

- In the password file, instead of storing the original password only, the system stores additional $k-1$ ($k>1$) *honeywords* for each user's account. The original password and these *honeywords* are collectively known as ***sweetwords***.
- All the *sweetwords* should be very similar from the structural point of view, however, they should not be identical.

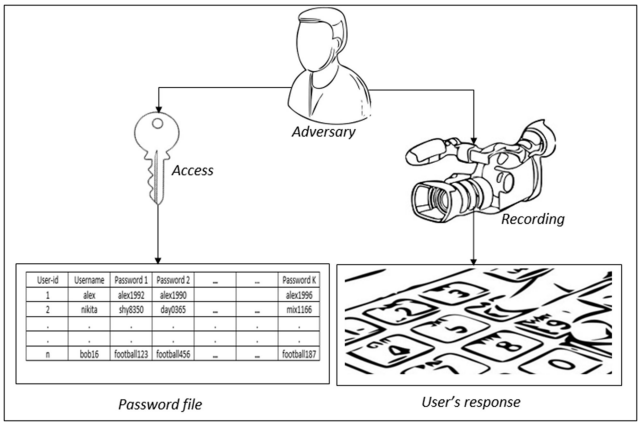


Fig. 2 *BST attack scenario in which server side attack and recording attack are simultaneously being conducted by the same adversary*

- Along with the username, the index position (varies between 1 and k) of the original password is stored in a different server, other than one storing the *sweetwords*. The server, maintaining these index values, is commonly known as ***honeyChecker***.

The aforementioned discussion suggests that HBAT works under the distributed security framework, which is much harder to compromise as a whole [36]. Following is an example of HBAT framework. In a system, let the username and password of a user be *alex* and *alex1992*, respectively. Along with the original password, *alex1992*, the system may maintain the following list of *sweetwords* in the password file for $k = 6$,

alex1990	alex1994	alex1991
alex1992	alex1995	alex1993

Along with the username, *alex*, the index of the original password, here four, is maintained in the *honeyChecker*.

Under this storage policy, if an adversary gets access to the password file, then it would not be possible for her to identify the original password, deterministically. Selecting a wrong password (or *honeyword*) as the original one detects the breach with the probability $(k-1)/k$.

3.3 Assumptions

In this paper, we do not emphasise to address the threat of password leakage due to both recording attack and the attack generated from a compromised password file. For example, let us assume that an adversary has the knowledge of all the *sweetwords*, maintained against a user's account in the password file. Now, in this situation, if the recording attack is performed during submission of response by the same user, then the eavesdropper can easily differentiate the original password from the list of *sweetwords*. We identify this threat as *Both Sided Threats* or *BST*. Though it is hard for an adversary to perform the *BST* attack in practice (as different resources need to be compromised at the same time), in theory, the impact of this threat is catastrophic. In Fig. 2, we depict this situation to which our scheme copes poorly.

Including the aforementioned one, we have the following few assumptions in this study:

- the adversaries are unable to launch the *BST* attack,
- distributed security is much harder to compromise as a whole,
- users perform the registration in a safe environment, free from any kind of threat.

4 Analysing the insecurity in existing n-SRRUAS

Here, we give a detailed analysis on why HBAT cannot directly be applied to the existing n-SRRUAS. In other words, we justify the statement in Note 8 (ref. to Section 1.2) under the scope of this section. The justification has been given based on the method of contradiction. Let an n-SRRUAS supports HBAT to store k *sweetwords* in the password file. To fit with the conventional HBAT, the n-SRRUAS *must always* meet the following criteria [17]:

- *Criterion 1:* The n-SRRUAS should always identify the proper *sweetword* for which the response, $\mathscr{R}$, is being submitted.
- *Criterion 2:* For k *sweetwords*, the probability of threat detection should always be $(k-1)/k$.
- *Criterion 3:* A genuine user should never get trapped while providing the correct $\mathscr{R}$. In other words, a correct response from the genuine user should not get mapped to any *honeyword*.
- *Criterion 4:* An adversary, giving $\mathscr{R}$ by targeting a *honeyword*, must not give an impression of providing the correct $\mathscr{R}$.

Let the ith $(1 \le i \le k)$ *sweetword* and its corresponding response set, R_S, be denoted as S_i and R_S^i, respectively. Based on (2), for a challenge $\mathscr{C}$, we can present R_S^i in the following form:

$$g_F : S_i \times \mathscr{C} \rightarrow R_S^i \tag{7}$$

As mentioned previously, for the response window R_W, containing all possible responses, each i, belonging to the set $\{1, 2, \ldots, k\}$, must satisfy the following condition:

$$R_S^i \subset R_W$$

Based on the aforementioned discussion, we propose the following lemma.

Lemma: If the size of response window be W and the average size of each response set be ΔW, then the probability of overlapping among the elements of the response sets will be

$$P_{\text{overlapping}} = 1 - \prod_{i=1}^{k-1} \frac{W - (i \times \Delta W)}{W}$$

where $k \times \Delta W \le W$ and, $k(>1)$ is the number of R_S formed on the response window, R_W.

Proof: We prove the above lemma based on the Roulette-Wheel selection strategy [37]. On the response window, corresponding to the sweetword S_i, the function $g_F(S_i, \mathscr{C})$ returns a response set, R_S^i. As $\mathscr{C}$ randomly varies in each session, therefore, the position of the response sets on the response window will also change depending on a session. Hence, the position of these response sets cannot be predicted early, before the beginning of a session. In a session, in the absence of any other response set on the R_W, the response set corresponding to the first sweetword (i.e. R_S^1) will not get overlapped with any other response set. However, the probability that the second response set (R_S^2) not overlapping with the first one (R_S^1) can be represented as follows:

$$\frac{(W - \Delta W)}{W}$$

Similarly, the probability associated with the third response set (R_S^3), corresponding to the third sweetword, would not overlap with the previous two response sets (R_S^1 and R_S^2) can be presented in the following manner:

$$\frac{(W - 2 \times \Delta W)}{W}$$

Therefore, on the response window, the probability of forming k non-overlapping response sets for k sweetwords can be equated as follows:

$$\frac{(W-\Delta W)\times(W-2\times\Delta W)\times\cdots\times(W-(k-1)\times\Delta W)}{W^{k-1}}$$

$$=\prod_{i=1}^{k-1}\frac{W-(i\times\Delta W)}{W}$$

Hence, the probability of overlapping among the elements of the response sets will be $1-\prod_{i=1}^{k-1}(W-(i\times\Delta W))/W$ (proved). □

For example, in S3PAS [12], $W=81$, $\Delta W=W/27$ and $k=6$, yield to $P_{\text{overlapping}}=0.45$. Also, for Asghar *et al.*'s proposal in [14], for $W=10$, $\Delta W=W/10$ and $k=5$, $P_{\text{overlapping}}$ can be calculated as 0.7. Fig. 3 pictorially depicts the overlapping among the response sets, for $k=6$ and $\Delta W \ll W$.

Fig. 3 shows that if $\mathscr{R}$ belongs to any overlapping area, then an n-SRRUAS cannot deterministically identify the corresponding response set and associated *sweetword* with it. This, in turn, leads to the following conclusive remarks.

Remark 1: From the submitted $\mathscr{R}$, belonging to the common region of multiple response sets, a system will not be able to identify the intended *sweetword* properly (*Criterion* 1 is not satisfied).

Remark 2: If the submitted $\mathscr{R}$ belongs to an overlapping region of $\overline{k}$ ($k\leq\overline{k}$) response sets, then chances of threat detection is reduced to $(k-\overline{k})/(k-\overline{k}+1)$. For example, if the submitted $\mathscr{R}$ falls in the region $R_S^1\cap R_S^2\cap R_S^6$ (see Fig. 3), then for $\overline{k}=3$ *sweetwords*, the probability of threat detection will be reduced from 5/6 to 3/4 (*Criterion* 2 is not satisfied).

Remark 3: Let R_S^i be the response set formed by the original password. Now, if the submitted $\mathscr{R}$ by a genuine user belongs to a common region of two response sets, R_S^i and R_S^j, then the system *may also interpret* that an adversary is submitting $\mathscr{R}$ by targeting the set R_S^j. For example, in Fig. 3, if $\mathscr{R}$ belongs to $R_S^1\cap R_S^2\cap R_S^6$, where we assume R_S^1 as the formed response set by the original password, then the system will also consider the possibility that an intruder is trying to login by targeting either R_S^2 or R_S^6 (*Criterion* 3 is not satisfied).

Remark 4: From a compromised password file, an adversary obtains all the *sweetwords*. Hence, based on $\mathscr{C}$, the adversary can identify the response set corresponding to each *sweetword*. For maximising her chances for login and minimising the probability of threat detection (see Remark 2), the adversary may adopt a smarter strategy. The adversary may submit a response which covers most of the response sets; e.g. $R_S^1\cap R_S^2\cap R_S^6$ or $R_S^2\cap R_S^4\cap R_S^6$ in Fig. 3. If any of these targeted response sets are formed from the original password (let it be R_S^1 or R_S^6), then along with considering a possibility of a threat, the system also interprets that the response is being generated based on the original password, $\mathscr{P}$, as $\mathscr{R}\in R_S^1$ (or $\mathscr{R}\in R_S^6$) (*Criterion* 4 is not satisfied).

Hence, we may conclude that for an n-SRRUAS, migration to HBAT is not immediate.

With the continuation of the above discussion, we make a *very important note* here. It might appear that – the server, containing the password file, may consult with the *honeyChecker* at the beginning of each authentication session for identifying the original $\mathscr{P}$. However, with some careful analysis, the readers can understand that this kind of setup does not fulfil the purpose of HBAT because of the following reason. If a server, storing the password file, brings the index of the original $\mathscr{P}$ from the *honeyChecker*, then the adversary, having access to that server, eventually gets to know the original $\mathscr{P}$ from the list of the *sweetwords*. Therefore, in our proposal, we have tried to modify the working principle of n-SRRUAS in such a manner so that k non-overlapping R_S can be formed on R_W to uniquely identify the intended *sweetword* from the submitted $\mathscr{R}$.

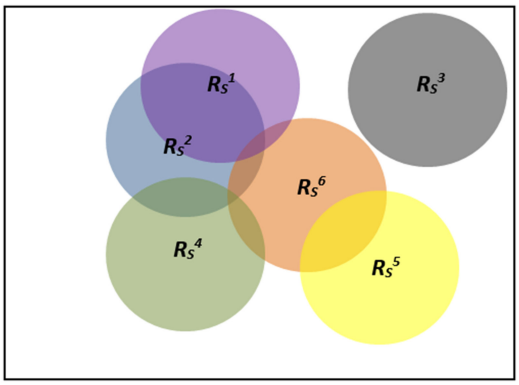


Fig. 3 *Overlapping among the response sets for $k=6$ sweetwords. R_S^i denotes a response set associated with ith sweetword. The rectangle in the above figure represents the response window (R_W)*

5 Proposed methodology

Before going into details – for overcoming the limitations discussed in the previous section, few factors have been identified at first which will play a crucial role in our study.

5.1 Identifying the factors

The factors here have been identified based on the working principle of n-SRRUAS. An n-SRRUAS accepts responses from a user in multiple rounds. In each round, the user and system use a portion of the password for generating the $\mathscr{R}$ and R_S, respectively. Based on this information, we define *Round Password* (*RP*), *Round Response* (*RR*) and *RR Set* ($(RR)_S$).

Definition 1: RP: RP is a part of $\mathscr{P}$ which is used by the functions $f()$ and $g_F()$ in each round for generating RR and $(RR)_S$, respectively.

Discussion 1: For $\mathscr{P}$ of length ℓ, we have observed that the length of *RP* varies between 1 and ℓ. For example, both $f()$ and $g_F()$ in S3PAS [12] always accept an *RP* of length lesser than ℓ. In contrary, both $f()$ and $g_F()$ in [14] accept the whole password as *RP*.

Definition 2: RR: With the help of RP and $\mathscr{C}$, the response generated by the user in each round is identified as RR.

Discussion 2: The above definition infers that *RR* in the ith round (identified as RR^i) can be represented in the form of $RR^i=f(RP^i,\mathscr{C}^i)$; where RP^i and $\mathscr{C}^i$ denote *RP* and $\mathscr{C}$, respectively, in the ith round.

Definition 3: $(RR)_S$: For the parameters RP and $\mathscr{C}$, we define $(RR)_S$ as the response set, which is returned by the function $g_F()$ in a round.

Discussion 3: The $(RR)_S$ in the ith round (identified as $(RR)_S^i$) can be expressed in the form of the following equation – $(RR)_S^i=g_F(RP^i,\mathscr{C}^i)$.

For the convenience of the readers, in Table 1, we have listed the defined terms so far. With the help of a hypothetical n-SRRUAS, next, we give examples of RP, RR and $(RR)_S$.

The reasons behind considering this hypothetical n-SRRUAS are following:

i. most of the n-SRRUAS follow complex login procedure,
ii. the hypothetical system follows a simple login process which is easy to understand,
iii. the hypothetical system obeys all the principles of an n-SRRUAS.

Table 1 Meaningful notations

Notation	Meaning	Notation	Meaning
$\mathscr{P}$	password	RP^i	RP in the ith round
$\mathscr{R}$	response	RR	round response
R_S	response set	RR^i	RR in the ith round
$\mathscr{C}$	challenge	$(RR)_S$	round response set
$\mathscr{C}^i$	$\mathscr{C}$ in the ith round	$(RR)_S^i$	$(RR)_S$ in the ith round
RP	round password	R_W	response window

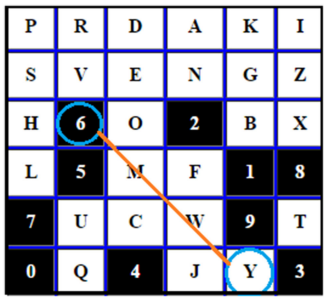


Fig. 4 *Visual interface of the hypothetical n-SRRUAS. A virtual line is connecting the characters 6 and Y in a round*

Hypothetical n-SRRUAS: We assume that the hypothetical system considers the set of all alphanumeric characters for forming both the visual interface and challenge space. At the beginning of each authentication session, the characters are randomly arranged in a 6×6 grid. The grid, in the form of a square matrix, is then displayed on the visual interface. Fig. 4 shows an instance of the visual interface of the hypothetical system.

Identifying RP: During registration, a user selects $\mathscr{P}$ of length 4 from the set of alphanumeric characters. Let the selected $\mathscr{P}$ be *6Y4Z*. The hypothetical system takes the responses from users in four rounds. Starting from the ith index, the user selects two consecutive characters of her password in the ith round. In the last round, the characters are chosen in a cyclic manner.

For example, for the considered $\mathscr{P}$ *6Y4Z*, RP in the first, second, third and fourth rounds will be *6Y*, *Y4*, *4Z* and *Z6*, respectively. In other words, we may also write this as $RP^1 = 6Y$, $RP^2 = Y4$, $RP^3 = 4Z$ and $RP^4 = Z6$.

Identifying $\mathscr{C}$: Randomly organised characters in the grid's cells can be considered as the system generated $\mathscr{C}$ for the considered n-SRRUAS. Therefore, the square matrix, as shown in Fig. 4, refers to an instance of $\mathscr{C}$. Like some of the existing state-of-the-art (e.g. proposed method in [12]), we assume that in this hypothetical system, generated $\mathscr{C}$ in each round remains the same.

Identifying RR: For generating RR in the ith round, the user first draws a virtual line on the visual interface by connecting the characters of RP^i. From the formed virtual line, the user then selects any character and inputs it to pass that round. For example, for RP^1 as *6Y* and the challenge shown in Fig. 4, RR^1 can be either '*M*' or '*W*'.

Identifying $(RR)_S$: For validating RR^i in the ith round, the system derives $(RR)_S^i$ from computing $g_F(RP^i, \mathscr{C})$. Therefore, as $\mathscr{C}$ shown in Fig. 4 and $RP^1 = 6Y$, generated $(RR)_S^1$ will be $\{M, W\}$. The user's response in the ith round is considered as valid one, only if, the system finds that RR^i is belonging to the set $(RR)_S^i$.

5.2 Proposed principles for incorporating HBAT

Based on the identified factors in the previous section, here we show how HBAT can be incorporated into an n-SRRUAS. For this, we have proposed four generic steps.

- *Step 1:* Given a $\mathscr{P}$, the system at first generates k *sweetwords* in such a manner so that for some particular $\mathscr{C}$, the generated R_S by the *sweetwords* are pairwise mutually exclusive.
- *Step 2:* For authenticating a user in a session, the system generates one of those particular $\mathscr{C}$ (mentioned in the previous step) such that with the help of the function $g_F()$, k *sweetwords* produce k mutually exclusive R_S.
- *Step 3:* Because of k mutually exclusive R_S, the generated $\mathscr{R}$, from a *sweetword* and $\mathscr{C}$, will not be a member of more than one R_S. Therefore, from R_S containing $\mathscr{R}$, the system can easily identify the corresponding *sweetword*.
- *Step 4:* System fetches the index of the identified *sweetword* and communicates it to the *honeyChecker* for further verification.

Though the above approach looks promising for protecting $\mathscr{P}$ by using HBAT framework, however, there is an associated issue which may threat the usability standard of such system. Let t_c be the required time for generating $\mathscr{C}$, producing k pairwise mutually exclusive (or non-overlapping) R_S by k *sweetwords*. As discussed earlier, most of the n-SRRUAS accept user's response in multiple rounds. Let us assume that in each round, the system generates $\mathscr{C}$ in such a manner so that k non-overlapping $(RR)_S$ can be formed. Therefore, an authentication system, accepting users' response in $n_r (> 0)$ rounds, increases the login time by $n_r \times t_c$. A significant increase in login time threats the usability standard, and therefore, we revise this strategy based on the following principles.

Principle 1: An n-SRRUAS must maintain k sweetwords in such a manner so that at least in a round, randomly selected by the system at the beginning of each authentication session, there exists no common element between any two, among the k number of $(RR)_S$.

Proof: Let us assume that in a round, i $(1 \le i \le n_r)$, randomly chosen by the system, any two $(RR)_S^i$ share some common elements. We denote the set of common elements as $\{\mathscr{E}\}$. Also, let $(RR)_S^i$ formed by the jth sweetword be denoted as ${}_j(RR)_S^i$. If generated $(RR)_S^i$ by the pth and qth sweetwords $(1 \le p, q \le k)$ share the set $\{\mathscr{E}\}$, then (8) stands.

$$ {}_p(RR)_S^i \cap {}^q (RR)_S^i = \{\mathscr{E}\} \tag{8} $$

Therefore, if RR^i belongs to $\{\mathscr{E}\}$, then the system understands that RP^i is selected either from pth or qth *sweetword*. However, the system cannot understand that between these two *sweetwords*, which one was selected by the user for generating the RR^i. Hence, for identifying a *sweetword* uniquely from the user response, an n-SRRUAS must satisfy Principle 1. □

In the ith $(1 \le i \le n_r)$ round, determined by the system at the beginning of an authentication session, (9) summarises the requirements for satisfying Principle 1.

$$ {}_p(RR)_S^i \cap {}^q (RR)_S^i = \varnothing, \quad \forall p, q \in \{1, 2, \ldots, k\} \text{ and } p \ne q \tag{9} $$

Before going into further discussion, next, we define a term, *all-different sweetwords*, which will be helpful for understanding Principle 2.

Definition 4: All-different sweetwords: Two sweetwords are said to be all-different if there exist no common characters between them.

For example, though *ABCD* and *PQRS* are all-different *sweetwords*, *ABCD* and *AEFD* are not.

Principle 2: An n-SRRUAS must choose k sweetwords in such a manner so that the following condition holds. In each round, the portion of the sweetwords, participating in a forming k number of $(RR)_S$ in that round, must be all-different from each other.

Proof: In the ith round, let ${}_jRP^i$ and ${}_j(RR)_S^i$ denote RP^i and $(RR)_S^i$, respectively, for the jth sweetword. For identifying a

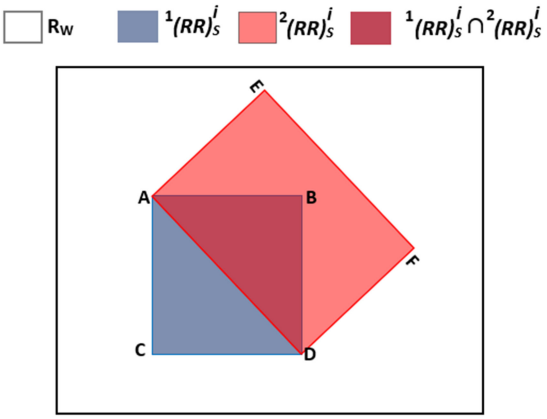


Fig. 5 *Unavoidable common zone for the response sets, generated from the sweetwords ABCD and AEFD. This is an example motivated from the n-SRRUAS, convex-hull-click in [9]*

sweetword uniquely from the user's response in the ith round, k non-overlapping $(RR)_S^i$ must be generated. For generating k number of non-overlapping $(RR)_S^i$, the following equation must hold:

$$g_F({}^1RP^i, \mathscr{C}^i) \neq g_F({}^2RP^i, \mathscr{C}^i) \neq \cdots \neq g_F({}^kRP^i, \mathscr{C}^i) \quad (10)$$

As $\mathscr{C}^i$ remains same in the ith round, therefore, (10) can only be satisfied based on the following equation:

$${}^1RP^i \neq {}^2RP^i \neq \cdots \neq {}^kRP^i \quad (11)$$

Now, as i is randomly chosen by the system at the beginning of an authentication session, therefore $\forall i \in \{1, \ldots, n_r\}$, both (10) and (11) must be satisfied. Notably, the above equation partially satisfies the criteria for generating k non-overlapping $(RR)_S^i$, as different sweetwords do not always guarantee non-overlapping $(RR)_S^i$. For example, motivated from [9], for two sweetwords ABCD and AEFD, the formed $^1(RR)_S^i$ and $^2(RR)_S^i$ (in the form of a rectangle) always share a common area on the R_W. Fig. 5 depicts this scenario. Hence, by storing all-different sweetwords, an n-SRRUAS must meet Principle 2 for incorporating HBAT. □

Principle 3: Depending on an n-SRRUAS, the value k varies between two and the number of possible non-overlapping $(RR)_S$ in a round.

Proof: For confusing an adversary, an HBAT needs to store at least one honeyword along with the original $\mathscr{P}$. Thus, the minimum value of k must always be 2. Principle 1 indicates that from a user's response, only non-overlapping $(RR)_S$ help an n-SRRUAS to identify the sweetword, which is being treated as the original $\mathscr{P}$ during the login. As each $(RR)_S$ is formed by an RP, which is nothing but the part of its corresponding sweetword, therefore, the value of k cannot exceed the number of possible non-overlapping $(RR)_S$. □

In Section 8, we have shown that these three principles are sufficient for incorporating HBAT to existing state-of-the-art.

6 Security analysis

It is quite straightforward that after adding *honeywords* to its password file, an n-SRRUAS inherits all the security properties of HBAT. However, incorporating HBAT may also influence the basic security standard of the n-SRRUAS (i.e. security standard against the recording attack). In this section, we discuss all these important security aspects of an n-SRRUAS, after incorporating HBAT into it.

6.1 Basic security properties: security against the recording attack

As discussed in Section 1.2, the obtained $\mathscr{C}$ and $\mathscr{R}$ from the recording evidence help an adversary to derive a set $\mathbb{S}$ (the probable candidates for $\mathscr{P}$) in each session. Along with the other alternatives, the set $\mathbb{S}$ always contains the original $\mathscr{P}$ of the user. Therefore, following (5), an n-session ($n > 1$) resilient protocol against the recording attack discloses the $\mathscr{P}$ in the following manner:

$$\mathbb{S}_1 \cap \mathbb{S}_2 \cap \cdots \cap \mathbb{S}_n = \mathscr{P}$$

As suggested in [6], this value of n can also be expressed in terms of the following equation:

$$n = \frac{\text{Password space}}{\text{Average cardinality of } \mathbb{S} \text{ after each recorded session}} \quad (12)$$

From the above equation, it is not hard to see that if the proposed modification does not affect the *password space* and cardinality $\mathbb{S}$, then the session resiliency of an n-SRRUAS would not be decreased. To be precise, for not affecting the session resiliency, the HBAT protected n-SRRUAS must not also make any impact on $\mathscr{C}$ and $\mathscr{R}$, determining the cardinality of $\mathbb{S}$. Next, we have made a detailed discussion on the role of adversary, generating $\mathbb{S}$.

We denote $\mathbb{W}$ as the window which holds the $\mathscr{C}$ in a session. If $\mathbb{W}$ contains x elements in it then this can be represented by an $\boldsymbol{x}$ length vector. For generating $\mathscr{R}$, if a user uses a set of $\boldsymbol{y}$ ($\boldsymbol{y} < \boldsymbol{x}$) elements from $\mathbb{W}$, then that set can also be represented by a $\boldsymbol{y}$ length vector. Let $\boldsymbol{V}_y^t$ be the tth vector of length y. Now one can form $\binom{x}{y}$ possible $\boldsymbol{y}$ length vectors from $\mathbb{W}$. In Algorithm 1 (see Fig. 6), we have shown the conventional cryptanalysis (brute force method) executed by the adversary for deriving the probable candidate set $\mathbb{S}$ in a session.

The *compute*() function in this algorithm performs some computation on the subset of $\mathbb{W}$, which is $\boldsymbol{V}_{\boldsymbol{m}}^{\boldsymbol{i}}$. The *compute*() function is correlated to the verification procedure adopted by an n-SRRUAS and hence, it varies based on the working principle of the n-SRRUAS. Readers can notice that if the proposed modification does not alter the values of $\boldsymbol{x}$ and $\boldsymbol{y}$, then the time complexity of the cryptanalysis remains unchanged. Moreover, if the farrago of HBAT and n-SRRUAS does not impact on the working principle of the n-SRRUAS, then the nature of the *compute*() function is expected to be unaltered. Not affecting the parameters $\boldsymbol{x}$, $\boldsymbol{y}$ and function *compute*() ensure that the denominator of (12) too remains unaffected. The aforementioned discussion infers that an efficient amalgamation of HBAT and n-SRRUAS hardly degrades the value of n.

Algorithm 1: generate_$\mathbb{S}$ ($\mathbb{W}$, $\mathcal{R}$)

for ($i := 1$ *to* $\binom{x}{y}$) **do**
  if ($\boldsymbol{V_m^i} \in \mathbb{W}$ && *compute*($\boldsymbol{V_m^i}$) $\vdash \mathcal{R}$) **then**
    $\mathbb{S} \leftarrow \boldsymbol{V_m^i}$;
return($\mathbb{S}$);

Fig. 6 *Algorithm 1: generate _$\mathbb{S}$ ($\mathbb{W}$, $\mathcal{R}$)*

6.2 HBAT security properties

There are three well defined security parameters related to an HBAT [17].

i. security against DoS,
ii. flatness,
iii. security against *multiple system vulnerability* (MSV).

However, some salient research in this domain [38] does not consider the third parameter as the significant one. This is because MSV occurs due to the breach in the distributed security framework and, HBAT itself works under the assumption that distributed framework is much harder to compromise as a whole. Hence, in our discussion too, we have only considered the first two security parameters.

Security against DoS: Having the knowledge of the original $\mathcal{P}$, if an adversary can guess any of the *honeywords* without compromising a password file, then she may intentionally submit that *honeyword* to make system understand that the password file has been compromised when it is actually not. Under this situation, a system denies any further login attempt from the users for security reasons. In this way, the adversary becomes successful in triggering the DoS attack. Handling this threat is extremely important given the fact that after recording n sessions, an adversary gets to know $\mathcal{P}$ which, in turn, allows her to mount the DoS attack if the *honeywords* are easily guessable.

For providing security against DoS, a system may adopt

- strong security policy against DoS,
- light security policy against DoS [38].

Under the strong security policy, once the system senses submission of a *honeyword* for any user's account, it blocks all the registered users' account in the password file. In contrast, under the light security policy, the system only blocks that user's account, for which submission of a *honeyword* has been detected.

Clearly, strong security policy against the DoS attack is much more sensitive than the weak security policy. Hence, a system, governed by the strong security policy, must ensure that knowledge of the original $\mathcal{P}$ provides little chance for guessing any *honeyword*. In order to achieve this, an n-SRRUAS must obey the following rules:

- *Rule* 1 *for avoiding DoS under the strong security policy:* Under the guidance of Principle 2, an n-SRRUAS may choose the *honeywords* using modelling-syntax-approach, proposed in [39].
- *Rule* 2 *for avoiding DoS under the strong security policy:* An n-SRRUAS must choose the significantly lesser value of k compared to its maximum limit.

However, for an n-SRRUAS, providing weak security against the DoS attack, a system may adopt light security policy for handling the threat [38].

Flatness: An ideal HBAT must generate a list of k equally probable *sweetwords* for each user's account so that an attacker does not have any advantage in discarding the *honeywords* from the compromised password file. Therefore, for a perfectly flat system, the probability of choosing the original password always remains $1/k$. As modelling-syntax-approach in [39] is capable of generating flat lists of *sweetwords* (until there exist a correlation between the username and password), therefore, governed by Principle 2, an n-SRRUAS may use the proposed approach by Bojinv *et al.* for generating the *honeywords*.

7 Usability analysis

Like the security analysis, usability analysis of an n-SRRUAS (after incorporating HBAT) also covers both the basic and as well as HBAT usability features.

7.1 Basic usability properties: login time and error rate

Login time and error rate in an n-SRRUAS is mainly influenced by the login complexity of the method. Roughly speaking, the login complexity is determined by the cognitive overload on the users during the authentication procedure. In case of an n-SRRUAS, the login complexity mainly signifies the amount of effort a user must give for deriving $\mathcal{R}$ from $\mathcal{P}$ and $\mathcal{C}$.

In [11], the authors' contribution showed that it is possible to measure this user effort theoretically. In our contribution too, we have emphasised this to identify the factors, influencing the login complexity of n-SRRUAS. The theoretical analysis here is mainly driven by following two components, *Cognitive Workload* ($\mathbb{CW}$) and *Memory Demand* ($\mathbb{MD}$).

Cognitive overload: $\mathbb{CW}$ is measured by the total reaction time (in seconds) for each atomic cognitive operation. There are four well defined atomic cognitive operations associated with a human identification protocol.

- (single/parallel) recognition [40],
- (free/cued) recall [41],
- (single-target/multi-target) visual search [42],
- simple cognitive arithmetic [43].

Recognition allows a user to compare the presented elements (may be $\mathcal{C}$) with $\mathcal{P}$, stored in memory. According to one of the most well-known recognition models, presented in [40], the total reaction time ($\mathbb{RT}_\alpha$) for this operation can be presented in the form of the following equation:

$$\mathbb{RT}_\alpha = \begin{cases} 0.3964 + 0.0383 \times |\mathcal{P}_M| & \text{single recog.} \\ (0.3964 + 0.0383 \times |\mathcal{P}_M|) \times \lceil I_d/4 \rceil & \text{parallel recog.} \end{cases} \quad (13)$$

where $|\mathcal{P}_M|$ ($\leq |\mathcal{P}|$) denotes the size of $\mathcal{P}$ memorised by a user to pass an authentication round. Notably, for an average user, the parallel recognition channels are limited to four and hence, if I_d items are displayed as $\mathcal{C}$, then the parameter $\lceil I_d/4 \rceil$ influences the formula of single item recognition [44], shown in (13).

Proposed n-SRRUAS in [16], the high-complexity CAS and low-complexity CAS are examples of single and parallel item recognitions, respectively.

The recall is another form of memory retrieval operation which is much slower than recognition. There are two basic variants of recall operation – free recall and cued recall. In free recall, a user is given a set of objects at first and then she is tested by recalling the objects in no specific order. However, in case of a cued recall, a set of objects are given to the user with some specific cues. In the test phase, the cues are provided to help the user in recalling the objects. It is not hard to see that cued recall is much easier than free recall and hence, most of the n-SRRUAS prefer cued recall over the free one for not degrading the usability standard. As reported

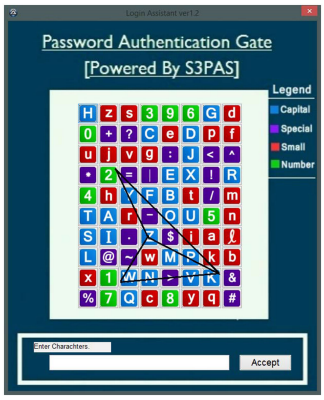


Fig. 7 *Visual interface of S3PAS for* $\mathbb{T} = 80$ *and virtually formed triangles by the user in the first two rounds for the password 2KZW. The formed virtual triangles (marked by black borders) in the first two rounds are Δ2KZ and ΔKZW. The valid responses may be '–' and 'M' for the virtual triangles Δ2KZ and ΔKZW, respectively*

by the authors in [41, 45], the reaction time for cued recall can be formulated as

$$\mathbb{RT}_{\beta} = (0.3964 + 0.0383 \times \sigma \times \gamma \times |\mathscr{P}_M|) \tag{14}$$

where σ (default value 1.969 [41]) is the ratio between cued recall and single item recognition. The parameter γ (default value 1.317 [45]) denotes the additional penalty for recalling the position of an element simultaneously.

Visual search operation occurs when a user tries to find some particular objects from a pool of objects. As suggested by the authors in [46], visual search by targeting a particular object among I_d objects demands significantly lesser reaction time compared to the involvement of multiple targets. The reaction time for single target visual search can be expressed in terms of the following equation [42, 46]:

$$\mathrm{RT}_{\gamma} = 0.583 + 0.0529 \times I_d \tag{15}$$

where I_d can be assumed as the number of elements displayed as $\mathscr{C}$.

Notably, some n-SRRUAS (e.g. proposed method in [7]) requires cognitive arithmetic operation during login from the user's end. Based on the experimental results in [43], the average reaction time for cognitive arithmetic operation varies between 0.738 and 0.959 s.

Memory demand: Like $\mathbb{CW}$, this is another important parameter which contributes to determining the user's effort from a theoretical perspective. For $\mathbb{MD}$ operation, the cost of each scheme can be expressed by the following equation:

$$\mathbb{MD} = \frac{|\mathscr{P}|}{\lambda_{\mathrm{op}}} \tag{16}$$

where λ_{op} stands for the accuracy rate of the corresponding memory retrieval operation within a fixed memorisation time.

Since recognition is much easier than recall, therefore, λ_{op} is evaluated as 29.6 and 84.8% for the recall and recognition, respectively [47].

Finally, the login complexity of an n-SRRUAS can be determined with the help of product with these two components, $\mathbb{CW}$ and $\mathbb{MD}$.

Therefore, after incorporating HBAT to an n-SRRUAS, if a user experiences no change in the login procedure (or values of different parameters influencing $\mathbb{CW}$ and $\mathbb{MD}$ do not get affected), then the error rate during login should remain unchanged. During incorporating HBAT to an n-SRRUAS, the designers should try to achieve this. However, as discussed in Section 5.2, to meet Principle 1, the login time will be increased by t_c. Section 8 shows that a careful bonding between HBAT and n-SRRUAS keeps t_c's value negligible.

7.2 HBAT usability properties

As suggested by the authors in [17, 48], the HBAT usability features mainly include

- system interference,
- stress on memorability,
- typo safety.

System interference: For generating the *honeywords*, if an HBAT interferes at the time of creation of the password during the registration phase, then there exists a system interference. An ideal HBAT should not interfere during the registration procedure. As modelling-syntax-approach has no system interference, therefore, without violating Principle 2, we suggest using this method for generating the *honeywords*.

Stress on memorability: For adopting the HBAT, if an n-SRRUAS forces the users to remember some additional information (or step) during login, then it imposes stress on users' mind. It is easy to relate that stress on memorability increases both the login time and error rate. Therefore, an HBAT protected n-SRRUAS should not put any additional stress on the users' mind.

Typo safety: An HBAT protected n-SRRUAS would also like it to be rare for a legitimate user to set off an alarm by accidentally entering a *honeyword*. Typos are one possible cause of such accidents. As modelling-syntax-approach is capable of generating quite different *sweetwords* from each other, thus it is preferable to use Bojinov *et al.*'s method for satisfying this usability criterion.

8 Application

To show that proposed idea can be applied to existing state-of-the-art for providing security against the server side attack, we have considered the S3PAS method, proposed by Zhao and Li in 2007 [12]. Notably, according to the report in [11], n-SRRUAS S3PAS can resist the recording attack for $n = 8$ sessions. *The other n-SRRUAS cannot be included here only because of space constraints.*

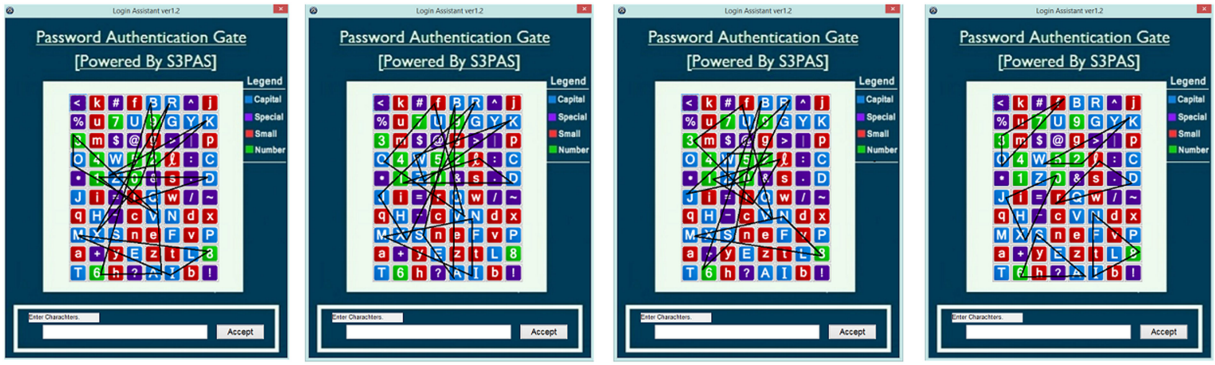
Incorporating HBAT into S3PAS: Under the scope of this study, we first show the basic working principle of S3PAS. Followed by this, with the help of the proposed principles in Section 5.2, we incorporate HBAT into this n-SRRUAS. The security and usability analyses of the revised S3PAS are then performed based on the discussions in Sections 6 and 7, respectively.

Authentication procedure in S3PAS: In S3PAS, a user chooses $\mathscr{P}$ of length 4 from the set of $\mathbb{T}$ elements. At the beginning of each authentication session, the n-SRRUAS randomly distributes these elements in an $m \times n$ grid. The grid is then displayed on the visual interface in the form of $\mathscr{C}$. In S3PAS, a user submits her responses in four rounds. To give her response in ith ($1 \le i \le 4$) round, the user performs the following tasks:

- the user first selects three consecutive characters from her $\mathscr{P}$, starting from the ith index,
- then she locates these characters on the visual interface (in the grid),
- thereafter, the user forms a virtual triangle by connecting those three characters,
- finally, a character inside that triangle is chosen as $\mathscr{R}$.

Let the chosen $\mathscr{P}$ of a user be *2KZW*. In Fig. 7, we have shown the visual interface of S3PAS in an authentication session. For $\mathscr{P}$ *2KZW*, the virtually formed triangles in the first two rounds are also shown in this figure.

Incorporating HBAT to S3PAS: As mentioned earlier, HBAT has been combined with S3PAS by the following the proposed principles in Section 5.2. We state that after merging HBAT with S3PAS, the HBAT secured S3PAS (or S3PAS-H) stores k *sweetwords* in its password file.



a *b* *c* *d*

Fig. 8 *Visual interface returned by S3PAS-H. For the sweetwords 2KZW, 8IMN, 6ABS, 0XRJ, 3OVf and rD1ℓ, this interface ensures that the generated triangles in the fourth round uniquely identify the user's response*

(a) Formed virtual triangles in the first round on the visual interface of S3PAS-H. The part of the *sweetwords* responsible for forming the triangles are *2KZ, 8IM, 6AB, 0XR, 3OV* and *rD1*, ***(b)*** Formed virtual triangles in the second round on the visual interface of S3PAS-H. The part of the *sweetwords* responsible for forming the triangles are *KZW, IMN, ABS, XRJ, OVf* and *D1ℓ*, ***(c)*** Formed virtual triangles in the third round on the visual interface of S3PAS-H. The part of the *sweetwords* responsible for forming the triangles is *ZW2, MN8, BS6, RJ0, Vf3* and *1ℓr*, ***(d)*** Formed non overlapping virtual triangles in the fourth round on the visual interface of S3PAS-H. The part of the *sweetwords* responsible for forming the triangles are *W2K, N8I, S6A, J0X, f3O* and *ℓrD*

Making use of Principle 1*:* In S3PAS, the elements of $(RR)_S$ can be considered as the characters belonging to a triangle. Therefore, for satisfying Principle 1, S3PAS-H must generate k non-overlapping triangles on the visual interface in the ith round, decided by the system at the beginning of an authentication session.

Making use of Principle 2*:* Readers can also notice that in the ith round, three consecutive characters (starting from the ith index) form RP in S3PAS-H. Therefore, for satisfying Principle 2, S3PAS-H ensures that in any round, part of the *sweetwords* – participating in forming the k triangles, must be all-different from each other.

Making use of Principle 3*:* Though Principle 3 helps an n-SRRUAS in determining the maximum value of k, it is always important to choose k′s value in such a way so that the n-SRRUAS builds robust security against the DoS attack. After investigating empirically, we have found that on average, a triangle in S3PAS (or S3PAS-H) contains 3.8 characters in it. Therefore, in a round, maximum 80/3.8 (or ∼21) non-overlapping triangle can be formed. Thus the value of k can be 21 at max. For building security against the DoS attack, we only consider six triangles (28.5% of possible triangles) or six *sweetwords* for protecting the original password.

For the chosen password *2KZW*, let the *honeywords* chosen by S3PAS-H are *8IMN*, *6ABS*, *0XRJ*, *3OVf* and *rD1ℓ*. It is easily verifiable that in each round, the part of *sweetwords* participating in forming the triangles is all-different from each other. As discussed in Section 6, we have used modelling-syntax approach for generating the *sweetwords*. Without the loss of generality, if S3PAS-H decides to form six non-overlapping triangles in the fourth round of an authentication session, then S3PAS-H must arrange $\mathbb{T}$ characters on the grid in such a manner so that the triangles $\Delta W2K$, $\Delta N8I$, $\Delta S6A$, $\Delta J0X$, $\Delta f3O$ and $\Delta \ell rD$ must share no common area. It is important to note that in other rounds, formed triangles may get overlapped. In Fig. 8, we have illustrated this scenario.

Impact of the modification: This kind of visual interface (or challenge) ensures that except in a particular round, determined by the system at the beginning of the authentication session, the responses from a user may belong to more than one triangles. However, in the predetermined round by the system, the response from the user only belongs to a single triangle as the formed triangles in this round are non-overlapping. Therefore, if S3PAS-H finds that in each authentication round, the response of the user is belonging to a particular triangle, formed by the part of a particular *sweetword*, then the system spots that *sweetword* and redirects its index value to the *honeyChecker* for further verification. In this way, with the probability 5/6 or 0.83, S3PAS-H can protect the original password against the server side threat.

Security analysis: The above text suggests that compared to S3PAS, S3PAS-H increases the security against the server side threat from 0 to 83.4%. As for both S3PAS and S3PAS-H, the nature of *compute* function and the values of $\boldsymbol{x}$ and $\boldsymbol{y}$ (see Algorithm 1 (Fig. 6)) remain unchanged, hence, (12) for both these methods yields the same value. Therefore, S3PAS-H allows no degradation of security standard while providing security against the recording attack. To reinforce the security level, chosen value of k here reduces the probability of DoS attack. Moreover, the *honeywords* in S3PAS-H are chosen by using modelling-syntax-approach for satisfying the flatness criterion.

Usability analysis: For login using S3PAS-H, as a user experiences no change in the login procedure, therefore, the error rate during the login is expected to be the same for both S3PAS and S3PAS-H. We can establish this fact from the theoretical point of view too. In both these methods, a user needs to locate three objects from a set of $I_d = \mathbb{T} = 80$ objects. As suggested by the authors in [49], the reaction time for the three-target visual search is ∼1.8 times longer than the single-target visual search on the same window. Therefore, $\mathbb{RT}_\gamma$ for both S3PAS and S3PAS-H yields to $(0.583 + 0.0529 \times 80) \times 1.8 = 8.667$. Also, based on (14) in Section 7, $\mathbb{RT}_\beta$ in each round for these two methods can be evaluated as $(0.3694 + 0.0383 \times 3 \times 1.969) \times (4/4) = 0.596$. From the presented text so far, $\mathbb{CW}$ in each round for both S3PAS and S3PAS-H results in $\mathbb{RT}_\beta + \mathbb{RT}_\gamma = 9.236$. Notably, $\mathbb{MD}$ for both these methods can be determined as $4/29\% = 13.51$. Therefore, for a login session of four rounds, the login complexity of both S3PAS and S3PAS-H will be $4 \times \mathbb{CW} \times \mathbb{MD} = 4.9 \times 10^2$. To be short, the theoretical analysis too suggests that the proposed modification does not impact on the users' effort during login.

However, we have observed that for constructing a visual interface (or $\mathscr{C}$) which can generate six non-overlapping triangles in a round, S3PAS-H takes an (average) additional $t_c = 1391$ ms. Hence, the login time in S3PAS-H is slightly increased by 1.3 s. As modelling-syntax-approach takes the responsibility for generating the *honeywords*, therefore, along with offering typo-safety, S3PAS-H imposes neither system interference nor stress on memorability.

Experimental analysis – An empirical study of the usability analysis: Theoretically, we already have shown that compared to S3PAS, S3PAS-H increases the login time (negligibly) by 1391 ms. However, to give a larger insight of our judgement about the usability aspect the proposed modification, we have performed an experimental analysis to re-establish the fact that added security does not degrade the performance of the users.

To conduct the experimental analysis, we took help from 27 participants, all having correct-to-normal eyesight. The participants were regular computer users and their age varied from 19 to 41. Among the participants, 21 were male and the rest were female. It is *important to note* that according to the report in [50], most of the studies in this area used a test group between 11 and 25, while very few used a test group of 50 or more participants [51]. Therefore,

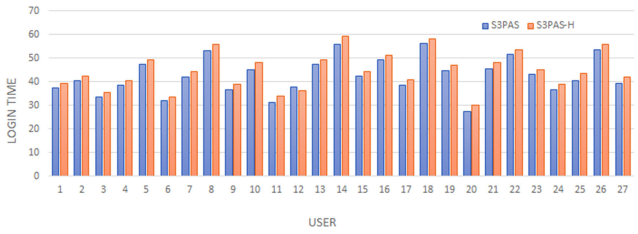


Fig. 9 *Participants performances in terms of login time for both the methods, S3PAS and S3PAS-H*

we believe that a set of 27 participants will provide an adequate base for our experimental study.

Experimental setup: At first, we used our laboratory's desktops (specifications – win7, 4 GB RAM, i3 processor) for uploading S3PAS and S3PAS-H. We demonstrated the login procedure for these methods to all the participants. Following this, based on the consent of the participants, we set the generic passwords '*2KZW*' for both S3PAS and S3PAS-H. For training purpose, we gave the participants 2 weeks' time for using these two methods. After 2 weeks, based on the request of three participants, we gave all the participants another 5 days for training them. Hence, after 19 days of the training period, we collected the test data in the following manner.

Collecting the test data for S3PAS and S3PAS-H: We asked each participant for login once using both the methods, S3PAS and S3PAS-H. Each method was consisting of four login rounds. We also requested them for continuing login until they successfully authenticate themselves using both these methods. In Fig. 9, we have shown login time achieved by the participants for both these methods.

Fig. 9 shows that in terms of login time, there is no significant difference between these two methods. For reassuring this fact, we performed the statistical significance test (or *ttest*) which yields to $(t(52) = 3.07 \times 10^{-12},\ p = 0.05)$ to conclude that there is no significant difference in the login time between the S3PAS and S3PAS-H. *It is important to note* that the obtained error rate for S3PAS and S3PAS-H were recoded as 10 and 12.9%, respectively.

Therefore, in a nutshell, we may claim that incorporating HBAT to S3PAS merely degrades the usability standard from both the theoretical and experimental point of views and thus holds the proposed guidelines in [19, 52].

9 Conclusion

In this work, we have identified the inability of n-session recording attack resilient unaided authentication systems for protecting the passwords against the server side threat. In this respect, we have pointed out that the *honeywords*, saviour of passwords in many occasions, cannot be directly applied to these systems due to their typical working procedure. In this paper, we have proposed three principles for incorporating *honeywords* to the targeted systems and show that these three principles are sufficient for achieving the desired objective. We have also applied the proposed idea to guard the password of an existing n-session recording attack resilient unaided authentication system. The outcome of our application shows that the password of the considered method can be masked successfully to resist the server side threat with very high probabilities (0.83). We believe that this paper shows utilisation of *honeywords* in a new direction which further benefits the security community from the usage of *honeywords*.

10 Acknowledgment

The authors thank the anonymous reviewers for their helpful comments and feedbacks which greatly help in improving the quality of this work.

11 References

[1] Pham, T.: 'Four years later, anthem breached again: hackers stole credentials', February 2015. Available at http://bit.ly/2btkw6K

[2] Fox-Brewster, T.: '13 million passwords appear to have leaked from 000webhost', October 2015. Available at http://bit.ly/1ReXjn7

[3] Raza, M., Iqbal, M., Sharif, M., *et al.*: 'A survey of password attacks and comparative analysis on methods for secure authentication', *World Appl. Sci. J.*, 2012, **19**, (4), pp. 439–444

[4] Kirda, E., Kruegel, C.: 'Protecting users against phishing attacks', *Comput. J.*, 2006, **49**, (5), pp. 554–561

[5] Weir, M., Aggarwal, S., De-Medeiros, B., *et al.*: 'Password cracking using probabilistic context-free grammars'. 2009 30th IEEE Symp. on Security and Privacy, Oakland, California, 2009, pp. 391–405

[6] Bai, X., Gu, W., Chellappan, S., *et al.*: 'PAS: predicate-based authentication services against powerful passive adversaries'. Annual Computer Security Applications Conf., 2008, ACSAC 2008, Anaheim, California, USA, 2008, pp. 433–442

[7] Hopper, N.J., Blum, M.: 'Secure human identification protocols'. Advances in Cryptology ASIACRYPT 2001, Australia, 2001, pp. 52–66

[8] Matsumoto, T., Imai, H.: 'Human identification through insecure channel'. Advances in Cryptology EUROCRYPT'91, Brighton, UK, 1991, pp. 409–421

[9] Wiedenbeck, S., Waters, J., Sobrado, L., *et al.*: 'Design and evaluation of a shoulder-surfing resistant graphical password scheme'. Proc. of the working Conf. on Advanced Visual Interfaces, Venezia, Italy, 2006, pp. 177–184

[10] Perković, T., Čagalj, M., Rakić, N.: 'SSSL: shoulder surfing safe login'. 17th Int. Conf. on Software, Telecommunications & Computer Networks, 2009, SoftCOM 2009, Hvar-Korcula, Croatia, 2009, pp. 270–275

[11] Yan, Q., Han, J., Li, Y., *et al.*: 'On limitations of designing leakage-resilient password systems: attacks, principles and usability', 2012

[12] Zhao, H., Li, X.: 'S3PAS: A scalable shoulder-surfing resistant textual-graphical password authentication scheme'. 21st Int. Conf. on Advanced Information Networking and Applications Workshops, 2007, AINAW'07, Ontario, Canada, 2007, vol. 2, pp. 467–472

[13] De-Luca, A., Hertzschuch, K., Hussmann, H.: 'Colorpin: securing pin entry through indirect input'. Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, Atlanta, GA, USA, 2010, pp. 1103–1106

[14] Asghar, H.J., Pieprzyk, J., Wang, H.: 'A new human identification protocol and coppersmith's baby-step giant-step algorithm'. Int. Conf. on Applied Cryptography and Network Security, Beijing, China, 2010, pp. 349–366

[15] Wu, T.S., Lee, M.L., Lin, H.Y., *et al.*: 'Shoulder-surfing-proof graphical password authentication scheme', *Int. J. Inf. Secur.*, 2014, **13**, (3), pp. 245–254

[16] Weinshall, D.: 'Cognitive authentication schemes safe against spyware'. 2006 IEEE Symp. on Security and Privacy, California, USA, 2006, pp. 6–11

[17] Juels, A., Rivest, R.L.: 'Honeywords: making password-cracking detectable'. Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security, Berlin, Germany, 2013, pp. 145–160

[18] Forouzan, B.A., Mukhopadhyay, D.: '*Cryptography and network security (Sie)*' (McGraw-Hill Education, Noida, Uttar Pradesh, India, 2011)

[19] Bonneau, J., Herley, C., Van Oorschot, P.C., *et al.*: 'The quest to replace passwords: A framework for comparative evaluation of web authentication schemes'. 2012 IEEE Symp. on Security and Privacy (SP), California, USA, 2012, pp. 553–567

[20] Gross, D.: '50 million compromised in Evernote hack. *CNN*', 2013

[21] Gaylord, C.: 'Linkedin, last. fm, now yahoo? Don't ignore news of a password breach', *Christ. Sci. Monitor*, 2012, July 13

[22] Herley, C., Florêncio, D.: 'Protecting financial institutions from brute-force attacks'. Proc. of the Ifip Tc 11 23rd Int. Information Security Conf., Milano, Italy, 2008, pp. 681–685

[23] Catuogno, L., Castiglione, A., Palmieri, F.: 'A honeypot system with honeyword-driven fake interactive sessions'. 2015 Int. Conf. on High Performance Computing & Simulation (HPCS), Amsterdam, Netherlands, 2015, pp. 187–194

[24] Takada, T.: 'Fakepointer: An authentication scheme for improving security against peeping attacks using video cameras'. The Second Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08, Valencia, Spain, 2008, pp. 395–400

[25] Kim, D., Dunphy, P., Briggs, P., *et al.*: 'Multi-touch authentication on tabletops'. Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, Atlanta, GA, USA, 2010, pp. 1093–1102

[26] Alsaiari, H., Papadaki, M., Dowland, P., *et al.*: 'Secure graphical one time password (gotpass): an empirical study', *Inf. Secur. J., A Glob. Perspect.*, 2015, **24**, (4–6), pp. 207–220

[27] Sun, H.M., Chen, S.T., Yeh, J.H., *et al.*: 'A shoulder surfing resistant graphical authentication system', *IEEE Trans. Dependable Secur. Comput.*, 2016, **15**, (2), pp. 180–193

[28] Spitzner, L.: '*Honeypots: tracking hackers*', vol. **1** (Addison-Wesley Reading, Boston, USA, 2003)

[29] Kim, G.H., Spafford, E.H.: 'Experiences with tripwire: using integrity checkers for intrusion detection', 1994

[30] Juels, A., Ristenpart, T.: 'Honey encryption: security beyond the brute-force bound'. Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, 2014, pp. 293–310

[31] Baddar, S.W.A.H., Merlo, A., Migliardi, M.: 'Anomaly detection in computer networks: a state-of-the-art review', *JoWUA*, 2014, **5**, (4), pp. 29–64

[32] Wang, D., Wang, P.: 'Two birds with one stone: two-factor authentication with security beyond conventional bound', *IEEE Trans. Dependable Secur. Comput.*, 2016

[33] Shen, C., Yu, T., Xu, H., *et al.*: 'User practice in password security: an empirical study of real-life passwords in the wild', *Comput. Secur.*, 2016, **61**, pp. 130–141. Available at http://dx.doi.org/10.1016/j.cose.2016.05.007

[34] Ma, J., Yang, W., Luo, M., *et al.*: 'A study of probabilistic password models'. 2014 IEEE Symp. on Security and Privacy (SP), San Jose, California, USA, 2014, pp. 689–704

[35] Gosney, J.: 'Password cracking hpc'. Passwords'12 Conf., Oslo, Norway, 2012
[36] Kontaxis, G., Athanasopoulos, E., Portokalidis, G.*, et al.*: 'Sauth: protecting user accounts from password database leaks'. Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security, Berlin, Germany, 2013, pp. 187–198
[37] Sher, A.M.: 'Enhanced roulette-style game'. Google Patents, 1998, US Patent 5,755,440
[38] Erguler, I.: 'Achieving flatness: selecting the honeywords from existing user passwords', *IEEE Trans. Dependable. Sec. Comput.*, 2016, **13**, (2), pp. 284–295. Available at http://dx.doi.org/10.1109/TDSC.2015.2406707
[39] Bojinov, H., Bursztein, E., Boyen, X.*, et al.*: 'Kamouflage: loss-resistant password management'. Computer Security–ESORICS 2010, Athens, Greece, 2010, pp. 286–302
[40] Sternberg, S.: 'Memory-scanning: mental processes revealed by reaction-time experiments', *Am. Sci.*, 1969, **57**, (4), pp. 421–457
[41] Nobel, P.A., Shiffrin, R.M.: 'Retrieval processes in recognition and cued recall', *J. Exp. Psychol., Learn, Memory Cogn.*, 2001, **27**, (2), p. 384
[42] Woodman, G.F., Chun, M.M.: 'The role of working memory and long-term memory in visual search', *Vis. Cogn.*, 2006, **14**, (4–8), pp. 808–830
[43] Campbell, J.I., Xue, Q.: 'Cognitive arithmetic across cultures', *J. Exp. Psychol., Gen.*, 2001, **130**, (2), p. 299
[44] Cowan, N.: 'The magical mystery four: how is working memory capacity limited, and why?', *Curr. Dir. Psychol. Sci.*, 2010, **19**, (1), pp. 51–57
[45] Corbin, L., Marquer, J.: 'Effect of a simple experimental control: the recall constraint in Sternberg's memory scanning task', *Eur. J. Cogn. Psychol.*, 2008, **20**, (5), pp. 913–935
[46] Woodman, G.F., Luck, S.J.: 'Visual search is slowed when visuospatial working memory is occupied', *Psychonomic Bull. Rev.*, 2004, **11**, (2), pp. 269–274
[47] Hogan, R.M., Kintsch, W.: 'Differential effects of study and test trials on long-term recognition and recall', *J. Verbal Learn. Verbal Behav.*, 1971, **10**, (5), pp. 562–567
[48] Chakraborty, N., Mondal, S.: 'On designing a modified-UI based honeyword generation approach for overcoming the existing limitations', *Comput. Secur.*, 2017, **66**, pp. 155–168
[49] Horowitz, T.S., Wolfe, J.M.: 'Search for multiple targets: remember the targets, forget the search', *Percept. Psychophys.*, 2001, **63**, (2), pp. 272–285
[50] Teh, P.S., Zhang, N., Teoh, A.B.J.*, et al.*: 'A survey on touch dynamics authentication in mobile devices', *Comput. Secur.*, 2016, **59**, pp. 210–235
[51] Kambourakis, G., Damopoulos, D., Papamartzivanos, D.*, et al.*: 'Introducing touchstroke: keystroke-based authentication system for smartphones', *Secur. Commun. Netw.*, 2016, **9**, (6), pp. 542–554
[52] Furnell, S.: 'Why users cannot use security', *Comput. Secur.*, 2005, **24**, (4), pp. 274–279