Secure Human-Computer Identification (Interface) Systems against Peeping Attacks: SecHCI

Shujun Li[1*] Heung-Yeung Shum[2]

[1] Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR, China
[2] Microsoft Research Asia, Beijing Sigma Center, No. 49, Zhichun Road, Hai Dian District, Beijing 100080, China

Abstract

This paper focuses on human-computer identification systems against peeping attacks, in which adversaries can observe (and even control) interactions between humans (provers) and computers (verifiers). Real cases on peeping attacks were reported by Ross J. Anderson ten years before. Fixed passwords are insecure to peeping attacks since adversaries can simply replay the observed passwords. Some identification techniques can be used to defeat peeping attacks, but auxiliary devices must be used and such devices are also insecure against peeping attacks if they are lost or stolen. Although more and more people get to know risks from peeping attacks, a practical solution has not been found.

This paper first gives a comprehensive review on peeping attacks and related issues, and then points out some basic design principles. Two general structures of secure human-computer identification systems are proposed against peeping attacks. A concrete SecHCI protocol and its various implementations are given, and a real Web service is developed for demonstration. The security and usability of the proposed protocol are investigated in detail. Although the usability of the proposed protocol is not yet sufficiently good, we believe that some design skills of the proposed protocol are useful for future work on SecHCI.

Keywords: secure human-computer identification (authentication), SecHCI, peeping attacks, shoulder-surfing attacks, observer attacks, pushing attacks, human iterative protocol (HIP), graphical passwords, human-computer interface (HCI)

1 Introduction

The task of human-computer identification[1] is to help a human (called *prover* or *claimant*) to prove its entity to a computer (called *verifier*), i.e. to help the *verifier* to distinguish the *prover* with the claimed identity from the malicious *impersonators*. The above identification is *unilateral* and can be extended to *mutual* one: both parties should prove their identities to another party. The *mutual* feature becomes very important in the situations that the verifier may be impersonated by adversaries.

There are many attacks aiming at different technique weaknesses of human-computer identification systems, such as dictionary attacks to fixed passwords [1]. This paper focuses on a special class of attacks: peeping attacks.

1.1 What are Peeping Attacks?

Peeping attacks are such attacks aiming at weaknesses lying between humans and computers: adversaries can observe all actions of humans on input terminals and interactions between humans (provers) and computers (verifiers), and even can disguise themselves as legal verifiers. Compared with other attacks on human-computer identification, much less attention is paid on peeping attacks and only a few research efforts have been made to study how to provide security against peeping attacks for present human-computer identification systems.

*The corresponding author, e-mail address: `hooklee@mail.com`, personal web site: `http://www.hooklee.com`.

[1] It is also called user authentication or verification in security literature [1].

Generally speaking, peeping attacks can be divided into two classes: 1) *passive (weak) peeping attacks* - adversaries can only passively observe the identification procedure of legal users, including all information that the users see and all responses that users make on the terminals; 2) *active (strong) peeping attacks* - adversaries can observe everything and also can impersonate verifiers to cheat legal users. Passive peeping attacks do not need special technical background, so anybody can carry out such attacks to grab our secrets. In security literature, passive peeping attacks have attracted some attention and are also called "observer attacks" [2–4] or "shoulder-surfing attacks" [5]. The simplest peeping "devices" are naked eyes, but hidden cameras are much more popular and indiscoverable [6, 7]. Malicious codes, hiddenly-deployed in computers, such as computer virus, Internet worms and Trojan horses, are rapidly increasing sources to run peeping attacks.

Generally speaking, it is much more difficult for an attacker to run active peeping attacks than passive ones, since he must sufficient experiences on how to hack a computer (the verifier) over the network. However, the prevalence of many powerful and user-friendly hack tools on Internet makes it much easier for a novice to try such attacks than before.

As a more technical method to carry out peeping attacks, compromising emanations[2] from computers are also useful for criminals equipped with TEMPEST devices [9, 10]. Because no physical access is needed for a TEMPEST-based peeping attack to a computer, so such peeping attacks are much more covert and dangerous than others. Additionally, it has been reported that optical TEMPEST from CRT monitors can also be used to recover some readable information [11, 12]. It is somewhat true that everybody is under surveillance everywhere and whenever [13].

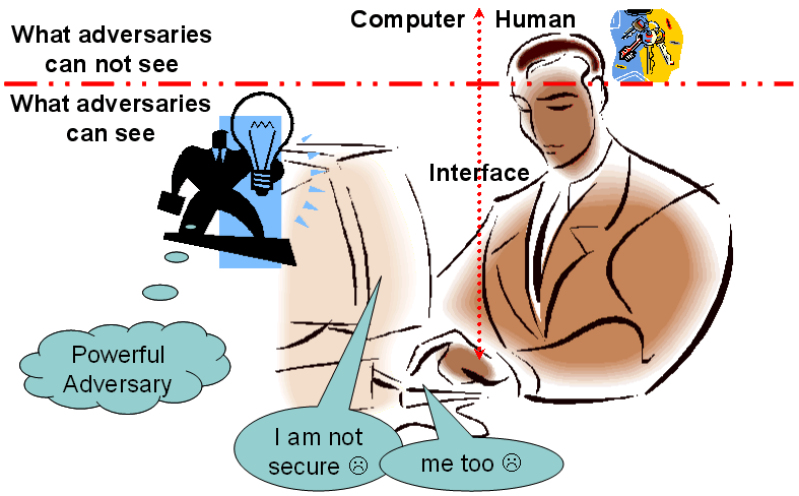


Figure 1: Peeping attacks: A graphical view

Fig. 1 gives a graphical view of peeping attacks in the real world. From the strictest viewpoint, all auxiliary input devices (such as keyboards, mouses, display monitors, USB disks, etc.) are untrustworthy; the used computer (CPU-s, hard disks, memory units, etc.; especially executable scripts/codes and security-sensitive data saved in hard disks or main memory) are untrustworthy; the remote server is untrustworthy; and even your facial expressions and eye movements are untrustworthy. Only mental ideas and computations occurring in your brain (i.e., human intelligence) is secure since it is almost impossible to covertly monitor brains and analyze the recorded brain signals with today's neuroscience technology.

1.2 Peeping Attacks in the Real World

Ten years before, a lot of real cases about peeping attacks have been collected and analyzed by [2, 3] to show the security problems with PIN-s of banking cards. Generally speaking, once an adversary successfully get the account

[2]Following the definition of US government [8], compromising emanations are unintentional intelligence-bearing signals which, if intercepted and analyzed, disclose the classified information transmitted.

number and the PIN-s of a customer's banking card, he can easily loot the customer's money with a counterfeit card. Here, we enumerate three typical ways to illegally get the account numbers and the PIN-s mentioned in Anderson's papers: 1) A cunning criminal can stand in an ATM queue, record others' PIN-s by observing their inputs on the ATM, and pick up discarded ATM tickets to get the corresponding account numbers; 2) A foxy maintenance engineer can covertly install specially-designed devices on ATM-s to collect a large number of account numbers and PIN-s; 3) The fastest growing way to get account numbers and PIN-s is to use bogus ATM terminals, for example, a group of criminals can set up fake vending machines to cheat honest customers. Apparently, the first two ways correspond to passive peeping attacks, and the third one is a typical active peeping attack. As live examples, for many ATM-s in Hong Kong (especially those deployed in public areas), there is a small post near the keypad, which generally says "*cover the keypad while you enter your PIN-s*" and "*if you detect anything unusual on the ATM, please report to our hotline immediately*".

In today's networked world, many banks encourage their customers to use online banking services to make e-finance over Internet. However, behind the great convenience and the claimed high security brought by such online e-services, good chances also come for bad guys to make peeping attacks: the trouble and risk to make counterfeit cards are cancelled, so they can grab your money much easily by only recording your online account and PIN-s with their hidden "eyes". Besides hidden cameras, computer virus, Trojan horses mentioned above, fraudulent or abducted web sites are also threatens such e-banking services. A real case was reported recently by HSBC (the Hongkong and Shanghai Banking Corporation Limited) [14]: a fraudulent web site `http://www.hkhsbc.com` was found to replicate the HSBC's Hong Kong web site `http://www.hsbc.com.hk` to try to steal credulous customers' online banking user ID-s and the corresponding passwords. In recent years, such frauds become more and more widely spread over Internet via so-called phishing attacks [15], in which the attacker generally sends a seemingly official electronic notification or message to a user and expects the user to expose his/her sensitive information, such as passwords, PINs, credit card number, etc., in a fake web site. Apparently, phishing attacks can be classified as special cases of active peeping attacks.

The great success of peeping attacks in the real world is due to the abuse of fixed passwords in most human-computer identification systems, since fixed passwords can not resist peeping attacks at all. Although some complex identification methods can resist passive peeping attacks, extra secure devices are required and such devices are generally sensitive to theft and loss[3]. Examples of such devices include one-time password generators [1, 16, 17], and smart cards to help users to make responses in challenge-response identification protocols [1, 18–21], and secret transparencies used in visual cryptography [22, 23], etc. Many identification methods that can resist passive peeping attacks cannot resist active peeping attacks at all. What about using biometrics to resist peeping attack? The answer is negative at present, since some recent public reports have shown that biometric technology nowadays is not so effective as expected in actual applications [24, 25]. In addition, privacy concerns and extra costs of biometric devices also limit the use of biometrics in many situations [26].

From the above discussion, we can see that new human-computer identification methods are wanted to provide acceptable security against peeping attacks (especially active peeping attacks). Formally speaking, the fight against peeping attacks is such a problem: *how a **naked** human can prove its identity to a **trustworthy** computer with **untrustworthy** input devices via an **insecure** channel?* In the above description, some words need further explanation:

- The word "***naked** human*" means that no any auxiliary device is available and users have to prove their identities with their mental intelligence;
- The word "***trustworthy** computer*" means that the right verifier is trustworthy for saving private information and the passwords;
- The word "***untrustworthy** input devices*" means that all interactions between users and the legal verifier are observable for an adversary;
- The word "***insecure** channel*" means that the communication channel is under control of an adversary who can deceive users into thinking he is the legal verifier.

[3] In some applications, such devices are further protected with fixed passwords, which makes the thing return back to the origin point.

In this paper, we pay our attention on practical solutions to peeping attacks, i.e. practical ***Sec**ure **H**uman-**C**omputer **I**dentification systems against peeping attacks* (SecHCI in short[4]). Because only a little work has been devoted to this topic till now, this paper wants to stir more future research on SecHCI by reviewing all related work known to the authors and by giving some new ideas to design SecHCI systems, in the hope that a new generation of passwords against peeping attacks will occur. We believe the new passwords will finally replace the small posts near the keypads of ATM-s, and provide higher security and better reliability to the future digital world.

The organization of this paper is as follows. In Sec. 2, we will introduce some related work on SecHCI. Sec. 3 introduces formal definitions on peeping attacks and SecHCI following the ones give in [30]. Sec. 4 discusses the problem how to design a practical SecHCI system, and some basic principles are proposed. Following the suggested design principles, two general structures against peeping attacks are given: Twins and Foxtail. In Sec. 5, a concrete SecHCI protocol is given, which is carefully designed following the suggested basic principles and the structure of Foxtail. The security and usability of the proposed Foxtail protocol are analyzed in detail, and the analysis establishes a paradigm for the evaluation of the overall performance of a given SecHCI protocol. In Sec. 6, two special topics related to SecHCI are discussed: eye-tracking and age verification technique, which are expected to play important roles in future research of SecHCI. The last section summarizes this paper.

2 Related Works

2.1 Partial Solutions

In almost all identification systems based on fixed passwords, a simple function against peeping attacks is adopted: displaying "******" or nothing on the screen when users type their passwords. It is the first firewall against peeping attacks and can provide a little security in practice.

Shielding plays the role of the second firewall. When we type passwords on the keyboard, the hands and the quick movements of the fingers may shelter the input pass-characters from the hidden "eyes" at a specific direction. When users input passwords (or make password-based responses to challenges) within a smaller space than keyboard, better shielding performance may be achieved, one example is the DAS graphical password used with hand-held devices [31]. Also, some banks have suggested their customers to cover the keypad when they enter the PIN-s on ATM-s. In addition, electromagnetic shielding is also a normal way to attenuate electromagnetic radiation and then to resist peeping attacks from compromising emanations [32], and optical shielding can be used to resist attacks based on optical TEMPEST techniques [11, 12]. Although shielding can provide acceptable performance against open peeping attacks, its performance against covert peeping attacks is poor. In addition, shielding can only resist passive peeping attacks, and cannot provide security against active peeping attacks.

The similar idea of shielding against peeping attacks has also been developed in visual cryptography, and a theoretically perfect solution called LVSVSS (Limiting the Visible Space Visual Secret Sharing Schemes) is proposed in [23]: with this technique, it is possible that only the legal user itself within a small visible space can clearly see the secret image generated from two overlapped transparencies. Using LVSVSS to generate one-time passwords is a good idea to resist peeping attacks, but the theft/loss-sensitive transparency is required as an auxiliary device so that humans are not ***naked***.

Besides the above partial solutions, there are still several other solutions with the use of challenge-response protocols and some specific algorithms, such as pass-algorithms [33], word associations [34] and Déjà Vu [4]. Basically, they are all one-time passwords with partial security against peeping attacks. The basic idea is to use long passwords and to expose only partial information about the whole password in each observed identification procedure. Obviously, such one-time passwords can only resist peeping attacks with a limited number of observations.

To provide satisfactory security against both passive and active peeping attacks, more specialized solutions are required. Unfortunately, compared with computer-computer identifications and human-computer identifications with the aid of trustworthy auxiliary devices, only a few efforts have been devoted to the design and analysis of such solutions [27, 28, 30, 35–39]. Till now, peeping attacks have still been out of sight of most cryptographers and cryptanalysts. In the following subsection, we will briefly survey specialized solutions against peeping attacks (for a more comprehensive survey, please refer to the authors' unpublished technical report [40]).

[4] In [27–29], other terms equivalent to SecHCI are used, such as HumanAut, HumanOIDs or PhoneOIDs. In this paper, we use the term SecHCI as a general word of such systems.

2.2 Specialized Solutions: A Brief Survey

To the best of our knowledge, the first published solution against peeping attacks was proposed in EuroCrypt'91 by [35], which is a challenge-response identification protocol employing redundant challenge objects to confuse users' password-based responses. Four years later, it was successfully cryptanalyzed in EuroCrypt'95 by [36]. It is shown that Matsumoto-Imai protocol can not resist active peeping attacks and the capability against passive peeping attacks were overestimated. Although Wang et al. suggested an improved version of Matsumoto-Imai protocol, its usability is so poor that it is impractical in real security systems.

Soon after Wang et al.'s cryptanalysis, Matsumoto proposed several new challenge-response protocols in [37] and [38], in which each user shares u v-length pass-vectors with the server and calculate dot products of the u pass-vectors and u challenge vectors to prove its identity. Although no cryptanalysis of Matsumoto protocols has been published, it has been pointed out [30, 39, 40] that none of the protocols are secure enough against peeping attacks, since an attacker can uniquely solve the secret password with $O(v)$ observed identifications (each one contains u dot products). For more details of the insecurity of Matsumoto protocols, please refer to our report [40]. Its security is similar to the above-mentioned one-time passwords against peeping attacks [4, 33, 34], and is lower than Matsumoto-Imai protocol proposed in EuroCrypt'91 [35].

Since the year 2000, the most promising advances about SecHCI against peeping attacks have been made in [30, 39, 41]. They gave some formal definitions on peeping attacks to clarify requirements and desired properties of a good solution to peeping attacks. Also, they presented several new challenge-response protocols with acceptable security to show some basic ideas about the design and analysis of SecHCI. However, the usability of proposed identification protocols is not very good so that they are impractical as human-executable protocols, especially for Protocol 2 proposed in [30] that can provide the capability against active peeping attacks.

Some further investigations (including some trivial errors and subtle neglected security problems) on Hopper-Blum protocols are given by [40]. One notable security problem is that the masquerading possibility of an attacker to guess right response is a little higher than the expected balanced value (1/2 for binary Protocol 1 and 1/10 for decimal version). This problem will be exponentially relaxed as the round number n increases, and satisfactory security can be reached when n is large enough.

As an actual implementation of Hopper-Blum protocols, an graphical SecHCI system was developed and available online at the web site of the CMU HumanAut Project http://www.captcha.net/humanaut. This system is a decimal prototype of Protocol 1 in [30] and is used to make user study on the design of future SecHCI protocols. Compared with the original protocol, the Web system has two specific features: 1) the password length is rather long ($k = n/2$); 2) the weight[5] of each generated challenge is not greater than $n/2 = k$, i.e., challenges are not randomly generated from all valid challenge vectors. The two features cause both usability and security problematic: 1) usability - since n must be large enough to ensure high security, the password will be too long for users to select and remember them; 2) security - because challenge vectors are not generated at random from all valid ones, the security problem about the masquerading probability will be enlarged (see Fig. 15 of [40]).

Besides the above work, there still were some efforts towards solutions against peeping attacks: in Prof. M. Blum's class of the course "CS 827: Security and Cryptography" [27], many kinds of possible solutions called PhoneOIDs[6] are proposed. All proposed PhoneOIDs have weak usability since the passwords are too long. In addition, it has been known that all proposed PhoneOIDs in the class are insecure [27, 42].

Till now, a really practical solution of SecHCI has not been found to settle the paradox between high security and acceptable usability. Based on experiences and lessons obtained from previous work on SecHCI, this paper will give some basic principles to design SecHCI systems, two general structures and a concrete SecHCI protocol as an example of one proposed structure. Detailed analyses show that the concrete protocol can achieve acceptable security, and that its usability is tightly related to implementation details. The results on the usability can be extended to other SecHCI protocols, and some facts are pointed out to make the usability better. Following the ideas and results given in this paper, it is expected that some better solutions will be found in future.

[5] Here, the ***weight*** of a binary challenge vector is defined as the number of 1s.

[6] PhoneOIDs are such protocols are carried by naked humans via phones.

3 Formal Definitions of Peeping Attacks and SecHCI

Before discussing how to design a SecHCI, let us firstly theoretically clarify the question "what is a SecHCI" and "what is a good SecHCI". In this section, we will give some formal definitions and concepts on peeping attacks and SecHCI. Most definitions and concepts are firstly introduced by [30] and further polished by [40].

3.1 Basic Definitions

To describe the definitions more clearer, we define a human-computer identification protocol (HCIP) as an interactive protocol between the following two parties: H (Human) with an auxiliary input x, and C (Computer) with an auxiliary input y. Denote the result of interaction(s) between H and C as $\langle H(x), C(y)\rangle$, and the transcript of information exchanged during their interaction by $T(H(x), C(y))$. Here, the auxiliary inputs mean the passwords shared between H and C, $T(H(x), C(y))$ means challenges and responses[7], and $\langle H(x), C(y)\rangle$ equals to `accept` or `reject`.

The basic task of C in a HCIP is to `accept` H if he can show his knowledge about a password z shared between H and C, and to `reject` H if he cannot show such knowledge. The above two sides of a HCIP is respectively called *completeness* and *soundness* in this paper. If a HCIP can be performed by a ***naked*** human without any auxiliary device, then we say this HCIP is *human executable*. The formal definitions of *completeness*, *soundness* and *human executability* are given as follows.

Definition 1 *A HCIP is **complete**, if for any auxiliary input z, $Pr[\langle H(z), C(z)\rangle = \texttt{accept}] \geq 1 - P_c$, where P_c is a small negligible probability.*

Completeness means that any legal user H can successfully prove its own identity to C with an overwhelming probability. Here, P_c denotes the ***work reliability*** of a HCIP.

Definition 2 *A HCIP is **sound**, if for any input pair $x \neq y$, $Pr[\langle H(x), C(y)\rangle = \texttt{accept}] \leq P_s$, where P_s is a small negligible probability.*

Soundness means that any user H can hardly (with a negligible probability) cheat C with others' identities. Here, P_s denotes the basic ***security*** of a HCIP. Generally, P_s is the maximal probability of an attacker to run online attacks under the condition that he has no any idea of the identification procedure of the target user. Apparently, soundness does not cover security against peeping attacks. Security against passive and active peeping attacks will be defined in the next subsection.

Definition 3 *A HCIP is **(α, β, τ)-human executable**, if any response $H(x)$ can be performed by a $(1-\alpha)$ portion of the human population (who can use their brains to make responses) with an error probability β, and can be completed within τ seconds.*

Good human executability means most humans can successfully prove themselves with high probability and within a short time. Here, α, β, τ are mean values of all capable humans and denote the ***usability*** of a HCIP. Please note β is essentially different from P_c: β denotes the probability that humans make unintentional wrong responses in each identification (which is determined by the computation complexity of making responses to specific challenges), but P_c denotes the probability that the verifier rejects a legal user with correct responses (which is determined by the inherent stability of the employed verifying algorithm).

Apparently, the ideal parameters of human executability are $\alpha = \beta = 0$ and $\tau = 1$, and HCIPs that humans cannot execute (i.e., the HCIPs requiring auxiliary devices) are $(1, 1, \infty)$-human executable. Practically speaking, a HCIP will be *nearly perfect* if its human executability is close to fixed textual passwords: $\alpha, \beta \leq 0.05$ and $\tau = O(10)$. Generally speaking, it is not very hard to get acceptable work reliability and security, thus the kernel of a practical SecHCI is how to realize acceptable usability, i.e., how to improve the human executability to an acceptable bound with subtle designs. Then what is the acceptable bound of human executability? From a user study reported in Sec. 2.2.2 of [40], we can see that the bound for most humans should be $(0.1, 0.1, 60)$.

[7] In an identification system based on fixed passwords, the challenge is null, and the response is the typed password.

3.2 Definitions about Peeping Attacks

Now let us give some definitions about peeping attacks, which are not covered by definitions defined in the last sub-subsection.

Definition 4 *A HCIP is* ***(p, k)-secure against passive peeping attacks****, if for any computationally bounded adversary $\mathcal{A}$,*

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

where $T^k(H(z), C(z))$ is a ***random*** *variant sampled from k* ***independent*** *transcripts $T(H(z), C(z))$. If p is independent of k, the HCIP is p-secure against passive peeping attacks.*

Definition 5 *A SecHCI is* ***(p, k)-secure against active peeping attacks****, if for any computationally bounded adversary $\mathcal{A}$,*

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

where $T^k(H(z), C(z))$ is a ***chosen*** *variant sampled from k transcripts $T(H(z), C(z))$. If p is independent of k, the SecHCI is p-secure against active peeping attacks.*

Only when an attacker can impersonate the legal verifier, it is possible for him to get a ***chosen*** variant $T^k(H(z), C(z))$. Apparently, a SecHCI with (p, k)-security against active peeping attacks is also (p, k)-secure against passive peeping attacks.

If a SecHCI protocol is (p, k)-secure against active peeping attack, we can call it ***(p, k)-secure against peeping attack*** in short. Here, please note that k has special significance in the security against peeping attack, because k need not to be cryptographically large but ***practically*** large enough to prevent peeping attacks. Consider the fact that you can only identify yourself $O(1000)$ times per year, $k = O(1000)$ is OK for a practically secure SecHCI.

3.3 SecHCI: A Formal Definition

Based on the definitions and concepts given in the last subsection, we can give a formal definition of SecHCI as follows.

Definition 6 (SecHCI) *A SecHCI is a HCIP satisfying the following properties: completeness, soundness, and (α, β, τ)-human-only executability with* ***acceptable*** *parameters, and* ***practical*** *security against passive (or active) peeping attacks.*

For a good SecHCI, besides the required properties, another feature is also desired: humans can detect the fake verifiers easily only by its own intelligence with considerable probability in one identification. This property is called ***human sensitivity (or consciousness) to active peeping attacks*** in this paper. *Human sensitivity to active peeping attacks* is useful to help users to find bad guys without exposing themselves and fight against them with active countermeasures.

To define the concept of human sensitivity to active peeping attacks, we follow the suggestion in [30] to add a third outcome to the interaction between H and C: $\langle H(x), C(y)\rangle = \perp$. Assume $C(z, \mathcal{A})$ denotes the fake verifier constructed by an adversary $\mathcal{A}$ who wants to get H's secret password z, $\langle H(z), C(z, \mathcal{A})\rangle = \perp$ if H detects the ongoing active peeping attack.

Definition 7 *A SecHCI is* ***(q, k)-human sensitive (or conscious) to active peeping attacks****, if for any computationally bounded adversary $\mathcal{A}$,*

$$Pr[\langle H(z), C(z, \mathcal{A}(T^k(H(z), C(z))))\rangle = \perp] \geq 1 - q,$$

where $T^k(H(z), C(z))$ is a ***random*** *variant sampled from k* ***independent*** *transcripts $T(H(z), C(z))$. If q is independent of k, the SecHCI is q-human sensitive to active peeping attacks.*

Here, (p, k)-security against active peeping attacks plus (q, k)-human sensitivity (consciousness) to active peeping attacks corresponds to **(p, q, k)-detecting against active adversaries** in [30].

4 How to Design SecHCI Systems?

4.1 How Peeping Attacks Work?

In the last section we theoretically discuss what is a SecHCI and what is a good SecHCI. Now let us study how peeping attacks work in human-computer identification systems and try to find some basic principles to design SecHCI systems against peeping attacks.

4.1.1 A Theoretical Model

Without loss of generality, let us assume a HCIP is a protocol with t challenge-response pairs

$$\{c_1, r_1\}, \cdots, \{c_i, r_i\}, \cdots, \{c_t, r_t\}.$$

If users make right responses for all t challenges, then $\forall i = 1 \sim t$, $r_i = f_i(c_i, P)$, where $f_i(c_i, P)$ is a function known by legal users with the secret password P. Following Kerckhoffs' principle [43], the formulas of $f_1 \sim f_t$ should be public (i.e., also known by attackers) and only P keeps secret.

In a successful peeping attack, assume an attacker $\mathcal{A}$ has observed s identifications, then $\mathcal{A}$ can get an equation system with $n = st$ equations as follows:

$$\begin{cases} f_1(c_1, P) = r_1 \\ \cdots\cdots \\ f_i(c_i, P) = r_i \\ \cdots\cdots \\ f_n(c_n, P) = r_n \end{cases}, \text{ where } f_i \in \{f_1, \cdots, f_t\}. \tag{1}$$

Assume the password contains k independent secret parameters $P_1 \sim P_k$, then theoretically the above equation system may be solvable when $n \geq k$. In the following we would like to give some examples to show how the above equation system is solved, i.e. how peeping attacks work.

4.1.2 Some Examples

Firstly, let us consider the simplest example - an identification system based on fixed passwords: $k = t = 1$, $f_i(c_i, P) \equiv P$. So $n = 1$ equation is enough to solve the equation system and get $P = r_i$.

For Déjà Vu graphical identification system proposed in [4], $k = \#(P) > 1, t = 1$, and $f_i(c_i, P) = r_i \subset P$. Thus, as n increases, the password P can be recovered by $\bigcup_{i=1}^{n} r_i$. Since challenges are randomly generated, probabilistically $n = O(k)$ are enough to reconstruct P.

For Matsumoto Protocols proposed in [37, 38], $k = uv$, $t = u$ and each challenge-response pair $(\boldsymbol{c}_i, r_i)$ corresponds to a deterministic linear equation $r_i = \boldsymbol{c}_i \cdot \boldsymbol{P}_i$, where $\boldsymbol{P}_i$ is the i^{th} element of the password $\boldsymbol{P}$ and $\boldsymbol{c}_i, \boldsymbol{P}_i$ are both v-size vectors. Apparently, with v independent linear equations, an attacker can uniquely solve $\boldsymbol{P}_i$. To solve $\boldsymbol{P} = \{\boldsymbol{P}_1, \cdots, \boldsymbol{P}_u\}$, probabilistically $n = O(v)$ identifications are enough.

To resist peeping attacks, Matsumoto-Imai Protocol [35] and Hopper-Blum Protocol 1 [30] employ different methods to introduce uncertainty into $r_i = f_i(c_i, P)$. However, the biased information contained in each challenge-response pair can be used to remove the intentionally introduced uncertainty by replaying a single challenges for many times (a kind of active peeping attacks). Once the uncertainty is removed, $r_i = f_i(c_i, P)$ becomes a deterministic equation and the equation system will be solvable when n is sufficiently large.

4.2 How to Frustrate Peeping Attacks: Some Principles

From the above discussion, we have known the solvability of the above equation system is the essential reason that peeping attacks work. To design a SecHCI against peeping attacks, a general way is to introduce ***uncertainty*** to make the equation system ambiguous and then unsolvable from probabilistic point of view. It should be guaranteed that the introduced uncertainty cannot be removed by statistical analysis based on probability difference. Under such a condition, attackers can only try to solve the password by removing the uncertainty via random guess, whose computation complexity can be designed to be cryptographically large to provide sufficient security.

In this subsection, we will give three basic design principles to design a practical SecHCI, with which it is possible to cut off the shortcut to get a deterministic equation system by observing one or more identifications of the target user. Besides the three basic principles, we will also introduce some points on how to provide human sensitivity to active peeping attacks and how to relax the paradox between security and usability.

4.2.1 Three Basic Principles: Time-Variant Responses, Uncertainty & Balance

Principle A: Time-variant responses Apparently, time-variant responses are essential to resist peeping attacks, otherwise attackers can simply replay the observed response (i.e. the fixed password) to impersonate others. According to time-variant responses, challenges are generally also time-variant[8]. Thus, a SecHCI should be a challenge-response protocol, in which users make time-variant responses to one or more (time-variant) challenges. The time-variant challenges can be generated with three methods [1]: *random numbers*, *sequence numbers* and *time stamps*. The kernel of the design of a SecHCI should be the problem how to generate tim-variant challenges that are sufficiently human executable, that is to say, users can make time-variant responses to the challenges easily only with their brains and their secret passwords.

Principle B: Uncertainty Uncertainty means blurring the equation system (1) obtained from observed identifications. Apparently, if the equation system becomes uncertain, it may be impossible to solve the password without cancelling the uncertainty. There are many ways to introduce uncertainty: intentional illegal challenges [35], intentional wrong response [30, 39], fuzzy responses [44–46], etc. Uncertainty can be exerted on any part of each equation in Eq. (1): c_i, r_i, and f_i. Because the introduction of uncertainty generally slows down legal users' responses, its negative influences on usability should be carefully considered.

Principle C: Balance As we mentioned above, if there exist biased probabilities on the introduced uncertainty, then it will be possible for an attacker to remove the uncertainty with statistical analysis. Therefore, it should be ensured that the introduced uncertainty is **balanced**. This property is neglected in Matsumoto-Imai Protocol [35] and Hopper-Blum Protocol 1 [30], so they cannot resist replay challenge attack (one of the simplest active peeping attacks). Furthermore, the balance property should be satisfied for both challenges and responses.

4.2.2 Human Sensitivity to Active Peeping Attacks

We have mentioned that it will be better if a SecHCI can provide high human sensitivity to active peeping attacks. How to achieve such a function? A simple way is to **make intentional wrong responses**, which makes the involved challenge-response protocol from a unilateral one to a mutual one. This method can be used as a simple tool to initiatively detect active attacks, since fake verifiers will always outcome "`accept`" even if a wrong response is made.

The above method requires that users takes the initiative always when they prove themselves to verifiers. However, users are always lazy to do so in almost conditions, which is the reason that many cryptosystems fail in practice [47–49]. As a result, making intentional wrong responses becomes a **passive** way to detect active peeping attacks. We need an **active** mechanism to explicitly remind users the dangers from active peeping attacks.

A possible way is to **distinguish fake verifiers from illegal challenges**. In a SecHCI, if legal challenges are generated dependently on users' passwords and not all challenges are legal, an adversary can only replay legal challenges observed in previous identifications, otherwise he will be detected with a positive probability. The probability is equal to the occurrence probability of illegal challenges in all fake challenges. If the ratio of the number of legal challenges to the number of all possible challenges is $\alpha \in (0, 1)$ and fake challenges are generated at random in all possible challenges, the detecting probability will be $1 - \alpha$. This fact makes active peeping attacks less dangerous, since an adversary has to observe sufficiently many legal challenges to run an effective active attack. Furthermore, for most SecHCI protocols, the same challenge corresponds to the same response, which means that active attacks retrogress into passive ones. Hopper-Blum Protocol 2 [30] adopts illegal challenges to achieve human sensitivity to active peeping attacks.

[8] Although it is possible to use fixed challenges (such as one-time passwords), users have to remember one or more previous responses to make the next response, which is rather inconvenient in practice.

4.2.3 Security vs. Usability

It has been found that there exists a trade-off between security and usability in SecHCI systems, and it is rather difficult to design a SecHCI with the usability close to fixed passwords. The following points are useful to achieve a better balance between usability and security.

Graphical/Visual passwords Recently, the idea of using human vision capability to realize visual/graphical passwords has been proposed as a novel solution to the problems of fixed textual passwords. Most visual/graphical passwords are based on the following cognitive fact: humans can recall pictures more easily and quickly than words (but generally with less accuracy) [31, 50]. Theoretical and experimental analyses have shown that graphical passwords can provide better memorability and higher practical security than textual passwords. Also, generally visual/graphical implementations of a challenge-response identification protocol can make users complete identification more quickly than textual implementations. Till now, most proposed visual/graphical passwords can be categorized into three types:

1. *selective pictures based passwords* – select pass-images from decoy images, such as Passface^{TM} [51–55] (selecting pass-faces from decoy faces), Déjà Vu [4] and Awase-E [56];
2. *point-and-click passwords* – find your pass-positions/objects in a picture and click them, such as PassPic [57] and the graphical "password windows" in Passlogix® v-GO^{TM} SSO system [50, 58, 59];
3. *drawing-based passwords* – draw your pass-strokes on a $m \times n$ grid, such as the DAS graphical passwords [31].

For more details about visual/graphical passwords, please see Sec. 5.1 of [40].

Some usability bounds The password length and identification time should not be too long. The user study on the security of fixed passwords reported in Sec. 2.1.2 of [40] has shown that **16** is the upper bound of password length, and **one minute** is the upper bound of the identification time. In addition, the size and the number of displayed information (texts and/or pictures) should not be too large: for applications in PDA-like devices, the display space are rather limited.

Using CAPTCHA to relax security against online attacks Automatic robots have been widely-used to exhaustively crack users' passwords. As an newly emerging technology, CAPTCHA [60, 61] is very useful to defeat robots running online attacks. CAPTCHA can ensure that only humans can run online attacks and robots are rejected. As a positive result, considering the low efficiency of humans[9], the security complexity against online attacks can be relaxed to be 2^{30}, which is much smaller than the cryptographically strong complexity $2^{60} \sim 2^{100}$ [1]. In many challenge-response identification protocols, such a relaxation will enhance usability dramatically. Generally speaking, there are two ways to incorporate CAPTCHA into a SecHCI system: 1) adding an extra CAPTCHA challenge in each identification round; 2) using CAPTCHA technology to transform each challenge into a CPATCHA challenge. The latter has better usability (less display space), but requires more computation load.

Multiple security levels The user study on security of fixed passwords and peeping attacks reported in Sec. 2 of [40] has shown that there exist different security levels in different security applications. For some applications (such as those on finance and privacy) the usability can be relaxed to enhance the security, but for some applications (such as games) the security can be relaxed to enhance the usability. If the security against limited ($O(n)$, n is not too much) observations is enough, it will be much easier to achieve acceptable usability by relaxing security. Also, if only security against open peeping attacks is needed, the design of SecHCI systems will become even further easier. Therefore, the application background of a SecHCI is very important to direct

[9] Assume a hacker can run 1 login trial per second, then the total number of login trials will be about 2^{25}.

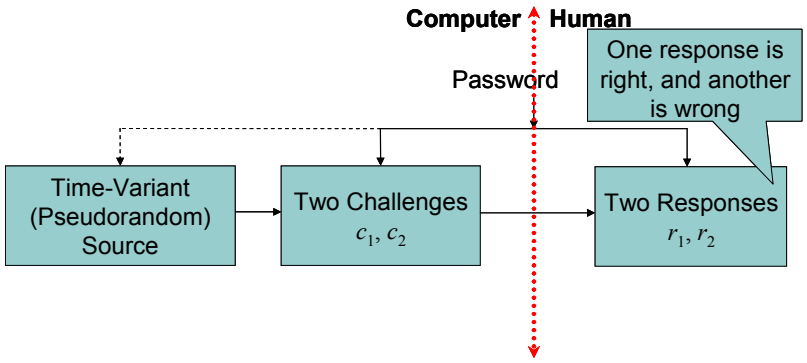


Figure 2: One identification round of a Twins protocol

4.3 Two General Structures of SecHCI Protocols

In this subsection, based on the principles proposed in the above subsection, we will give two general structures to design SecHCI systems. In this paper, they are respectively called Twins and Foxtail to emphasize the employed ideas to introduce uncertainty in Eq. (1). Basically speaking, it is expected that Foxtail has better overall performance on security and usability than Twins, but Twins is more concise and more useful to clarify the basic principles in the design of SecHCI protocols.

4.3.1 Twins

Twins is a general SecHCI structure to demonstrate the proposed principles in the last subsections. The identification procedure of Twins can be described as follows (see also Fig. 2):

$C \Rightarrow H$:	c_1, c_2 (two *independent* challenges)
$H \Rightarrow C$:	r_1, r_2 (two *related* responses)
$H \Leftrightarrow C$:	Repeat the above steps for t rounds
$C \Rightarrow H$:	If for each challenge pair c_1, c_2, one response of r_1, r_2 is correct and another is wrong, outcome `accept`, otherwise outcome `reject`

In the above SecHCI protocol, the right response r and the corresponding challenge c can connected with any function. That is to say, *any challenge-response HCIP can be extended to a Twins protocol against peeping attacks*. Here, we call the embedded challenge-response HCIP ***base protocol***. In the following, we briefly discuss the security and usability of Twins to show some requirements and extended versions.

Security against passive peeping attacks To prove themselves with a Twins protocol, users must randomly determine (via **private** and **balanced** coin-toss) which response should be intentionally wrong. If the coin-toss is really balanced and private, the success probability of guessing n right responses will be $2^{-n/2}$. That is to say, the probability to solve Eq. (1) will be $2^{-n/2}$ when n independent equations are observed. If the total number of secret pass-parameters n' is sufficiently large, the security against attacks based on random guess can be ensured. For digital computers nowadays, $n' \geq 150$ is required. Although 150 seems to be too large for humans to remember the passwords, it is possible to make it easy, for examples please refer to SecHCI protocols proposed in [30] and the one proposed in Sec. 5 of this paper.

Security against active peeping attacks In active peeping attacks, if there exist probability difference between the right response and the false response for a given challenge, an attacker can successfully get the right one by replaying a same pair of challenges for many times. To avoid such a risk, the probability to make the right response should be 0.5, i.e. users try to make a binary selection for each challenge: true or false. As a result, the $2t$ responses can be represented as a $2t$-bit number: $b_{1,1}b_{1,2}\cdots b_{i,1}b_{i,2}\cdots b_{t,1}b_{t,2}$, where $\forall i = 1 \sim t, b_{i,1} \oplus b_{i,2} = 1$. Such a requirement corresponds to another aspect of the **balance** feature.

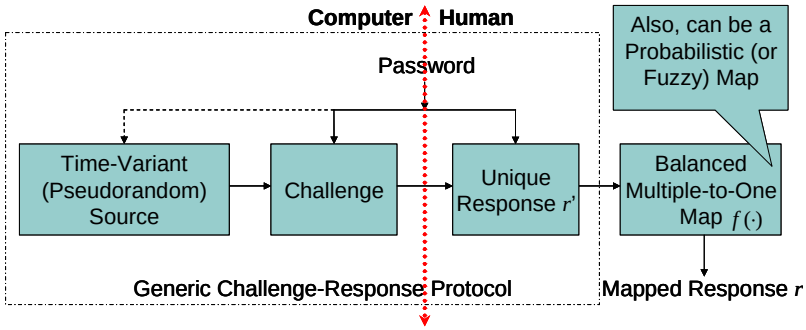


Figure 3: One identification round of Foxtail protocol

Human sensitivity to active peeping attacks A Twins protocol has an obvious disadvantage: it cannot provide human sensitivity to active peeping attacks. To achieve such a property, some modifications must be made on the original protocol. Here, we give one possible solution. Firstly, define some password-dependent challenges as a special set. When one or two challenges in this set appear in c_1, c_2, two responses r_1, r_2 in the current round should be both wrong. The identification will not complete until t normal challenge-response rounds are made. With the binary representation of Twins protocol, the $2t' \geq 2t$ responses can be denoted by $b_{1,1}b_{1,2}\cdots b_{i,1}b_{i,2}\cdots b_{t',1}b_{t',2}$, where $b_{i,1} \oplus b_{i,2} = 1$ holds for only t rounds among total t' ones. In other words, if all challenge-response rounds not satisfying $b_{i,1} \oplus b_{i,2} = 1$ are removed from $b_{1,1}b_{1,2}\cdots b_{i,1}b_{i,2}\cdots b_{t',1}b_{t',2}$, the binary form will return to its original version $b_{1,1}b_{1,2}\cdots b_{i,1}b_{i,2}\cdots b_{t,1}b_{t,2}$. Since faked verifiers do not know the password-dependent challenge set, then they do not know how many rounds there should be. As a result, it is possible for users to defect the existence of the faked verifiers with a positive probability.

Usability The half intentionally wrong responses make the identification time be about **twice** as large as that of the base protocol. What's worse, the private and balanced coin-toss further prolongs the identification time. Because it is generally hard for humans to make really random coin-toss, some methods must be used for assistance[10]. The employed coin-toss method must be carefully kept secret to avoid peeping attacks, so users should frequently change their coin-toss methods, that it to say, it is not advisable to use a fixed method for a long time. Additionally, to achieve human sensitivity to active peeping attacks, because more challenge-response rounds are required, the usability will be sacrificed to some extent. As a summary, the usability of Twins is not satisfactory.

4.3.2 Foxtail

Foxtail is another general SecHCI protocol with entirely different structure from Twins. Foxtail is proposed in hope that the unsatisfactory usability of Twins can be enhanced. Compared with Twins, Foxtail provides more flexibility on the implementation details and a better trade-off between security and usability with careful designs. In the next section, we will propose a concrete protocol to show Foxtail's overall performance.

The identification procedure of Foxtail can be described as follows (see also Fig. 3):

$C \Rightarrow H$:	c
$H \Rightarrow C$:	r
$H \Leftrightarrow C$:	Repeat the above steps for t rounds
$C \Rightarrow H$:	If $r = F(r_h) = F(f(c, P))$ holds for each challenge c, then outcome `accept`, otherwise outcome `reject`

Here, $r_h = f(c, P)$ means a **hidden response** to the challenge c, and $F(\cdot)$ is a multiple-to-one (or probabilistic, fuzzy) map and is called **Foxtail map**. The function of the Foxtail map is to introduce uncertainty into the response r, and it should be avoided that such uncertainty is removed by the composition of F and f (see Sec. 5.3.3 for

[10]For example, the second hand of a watch can be used to determine whether or not one should make the wrong response for the first challenge in each pair of challenges. Apparently, there exist a lot of available methods for us to use in practical identifications.

an example of such bad Foxtail maps). The Foxtail map can also be considered as a classifier, which transform the hidden response r_h into a specific class r. Here, each class should contain at least two elements to make the classifier $F(\cdot)$ be a multi-to-one map. Apparently, there should be at least 2 different classes, so r_h should have at least 4 different values. The Foxtail map $F(\cdot)$ should be carefully selected to avoid security weaknesses, since not all multiple-to-one maps can successfully cut off the way to solve Eq. (1).

In the above Foxtail protocol, the relation between the hidden response r_h and the corresponding challenge c can be any function. That is to say, *any challenge-response HCIP can serve as a* ***hidden base protocol*** *to construct a Foxtail protocol with any suitable Foxtail map* $F(\cdot)$.

Here, we only give a brief discussion on the security and usability of Foxtal protocols, and more details will be shown in the next section on a concrete Foxtail protocol.

Security Assume $F(\cdot)$ is a v-to-one map, the success probability to guess the exact value of r' is v^{-n} for n observed challenge-response pairs. That is to say, Foxtail protocol is $(v^{-n}, O(n))$-secure against passive peeping attacks. If the balance property and human sensitivity to active peeping attacks is ensured, Foxtail protocol will also be $(v^{-n}, O(n))$-secure against active peeping attacks. It implies that Foxtail can provide higher security than Twins. Of course, many special security issues should be carefully considered to analyze the security of a Foxtail protocol, in the next section we will propose a concrete Foxtail protocol to show how the security against peeping attacks is achieved with subtle details of its design.

Usability In Foxtail protocols, users need to make less (only about half) responses than they do in Twins. When $F(\cdot)$ is designed to be human executable with good parameters, Foxtail is expected to have similar usability to the base protocol (and better than Twins). Apparently, to optimize usability, the kernel task is how to find a good Foxtail map and a good base protocol with optimal usability.

5 A Foxtail Protocol and its Performance

In this section, we will propose a concrete protocol to show the performance of Foxtail as a practical solution against peeping attacks. Although the usability of this protocol is not yet sufficiently satisfactory, we believe that SecHCI protocols with better usability can be found following the basic ideas proposed in Foxtail and Twins protocols.

5.1 Description

Given a set $\mathcal{O}$ containing n different objects, assume the password P is a k-size subset of $\mathcal{O}$, where $k \geq 3$. For a subset $\mathcal{C} \subseteq \mathcal{O}$, its **similarity** with P (similarity in short) is defined as $\mathrm{Sim}_P(\mathcal{C}) = \#(\mathcal{C} \cap P)$, where $\#(A)$ denotes the size of the finite set A. Here, $\mathcal{O}$ is called the **objects pool** and $\mathcal{C}$ is called a **cell**. Based on the above definitions and notations, the identification procedure of the proposed Foxtail protocol is as follows:

$C \Rightarrow H$:	$\{\mathcal{C}_1, \mathcal{C}_2\}$, where $\#(\mathcal{C}_1) = \#(\mathcal{C}_2) = l \geq 3$
$H \Rightarrow C$:	r
$H \Leftrightarrow C$:	Repeat the above steps for t rounds
$C \Rightarrow H$:	If $r = \left\lfloor \frac{(\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)) \bmod 4}{2} \right\rfloor$ holds for each challenge, then outcome `accept`, otherwise outcome `reject`, where $\lfloor a \rfloor$ means the greatest integer not less than a

In the above Foxtail protocol, the hidden response

$$r_h = (\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)) \bmod 4,$$

and the Foxtail map $F(r_h) = \lfloor r_h/2 \rfloor$. To achieve the balance property, the two challenge-cells $\mathcal{C}_1, \mathcal{C}_2$ should be generated with two different methods: one is generated from all possible challenge-cells at random (this method is called **Ran-Rule**), and another is generated at random from all challenge-cells whose similarities are 0,1,2,3 (with the same probability $\frac{1}{4}$, this method is called **Uni-Rule**). To realize the balance property, n, k, l should also satisfy the constrain $3n = 2kl$. Before showing each challenge to users, all objects in the two challenge-cells

can be **randomly permutated** firstly, which makes it impossible for both users and adversaries to recover either challenge-cell from $2l$ shuffled challenge-objects. In Sec. 5.3, we will show that the random permutation is useful to guarantee the security against active peeping attacks.

5.2 Example Implementations

There are many possible implementations of the above Foxtail protocol. In this subsection we will give four ones of them to show how Foxtail protocols work in real situations. Actually, most graphical implementations can be incorporated into electronic games to attract users' interests and make the identification more funny, which may be useful to enhance the usability. From the four examples given here, we can see that a lot of good ideas are available for implementations of the proposed Foxtail protocol. We encourage readers to find more ideas for implementations with optimal usability in different applications of SecHCI protocols.

5.2.1 A Textual Implementation

In this implementation, the objects pool is a set composed of n characters. The following is one identification round of this example implementation.

Password: $P = \{\texttt{SecHCI}\}$
Challenge: $c = \{\texttt{aIi*.9jSiqe7xR}\}$ (Two hidden cells are $\mathcal{C}_1 = \{\texttt{aiIx9.R}\}$ and $\mathcal{C}_2 = \{\texttt{qe7*ijS}\}$)
Response: $r = \left\lfloor \frac{(\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)) \bmod 4}{2} \right\rfloor = \left\lfloor \frac{1 + 2 \bmod 4}{2} \right\rfloor = 1$
(the underlined characters are pass-objects)

5.2.2 An Icon/Image-Based Graphical Implementation

In this implementation, the objects pool is defined as a set composed of n different icons or images. It corresponds to the graphical passwords based on selective pictures [4, 51–56]. An icon-based implementation of the proposed Foxtail protocol has been developed and available online at http://www.hooklee.com/SecHCI as a WWW identification service. The default parameters are $n = 140, k = 14, l = 15, t = 20$, and n selected icons compose the objects pool $\mathcal{O}$. In Figure 4, we give one identification round of this identification system. Initial user study has shown that users can finish one identification process within 3 to 4 minutes. If the pass-objects are carefully selected by users to reflect their own interests (see so-called pass-rule discussed in Sec. 5.4.2 of this paper), the identification time can be further enhanced. Generally speaking, such a graphical SecHCI system is $(0.9, 0.1, O(180))$-human executable. In Figure 5, a snap of an actual identification process with the online systems is given.

Password:

Challenge (two 12-object cells are included):

Response: $r = \left\lfloor \frac{4 \bmod 4}{2} \right\rfloor = 0$ (the red circles enumerate the contained pass-objects)

Figure 4: An icon-based implementation of Foxtail: one identification round

5.2.3 A Map-Based Graphical Implementation

This implementation is based on a traffic map in a city (see Figure 6 for one identification round). The objects pool is the set composed of all streets between two neighbor crosses, including those from outside region to edge

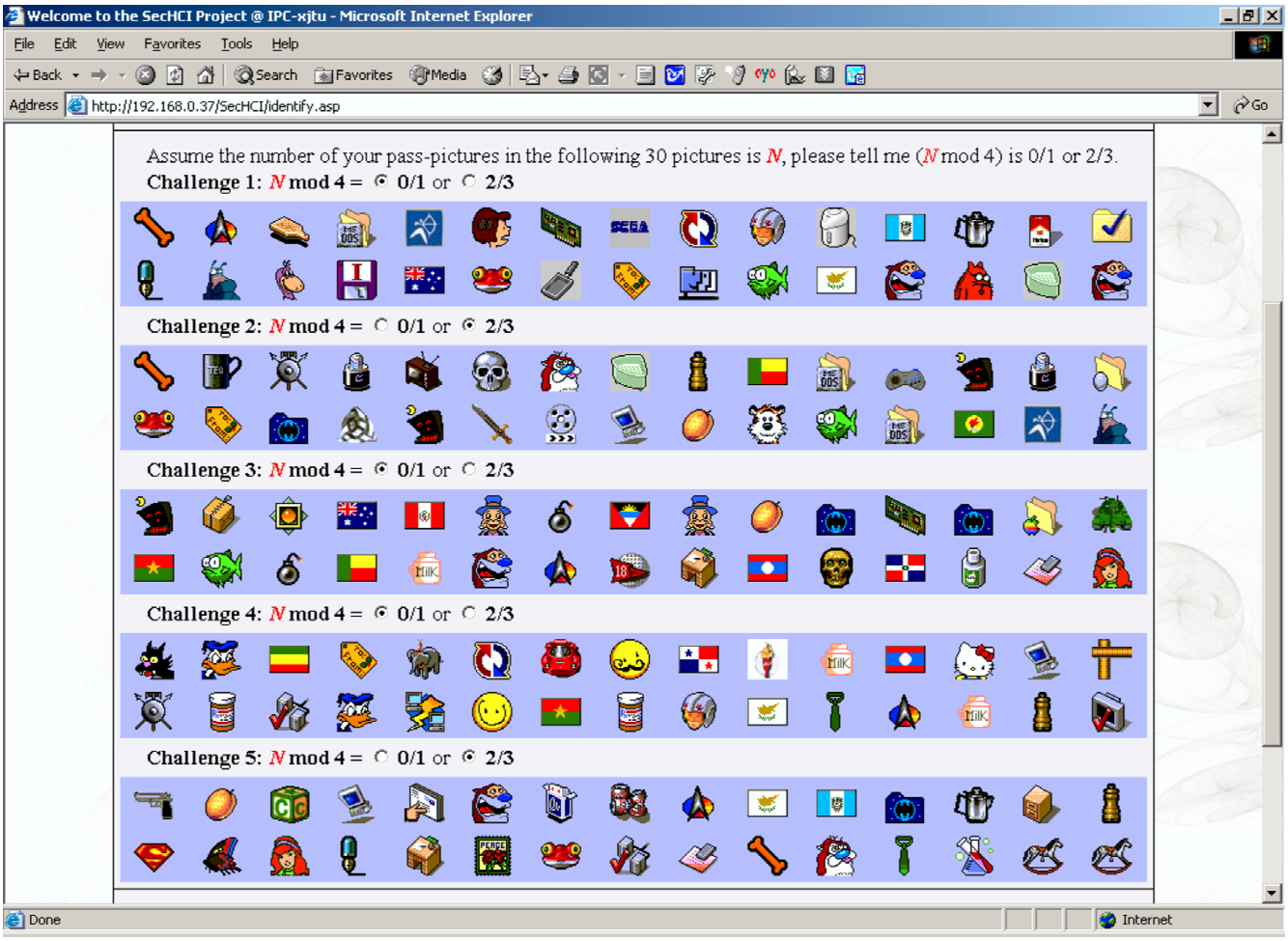
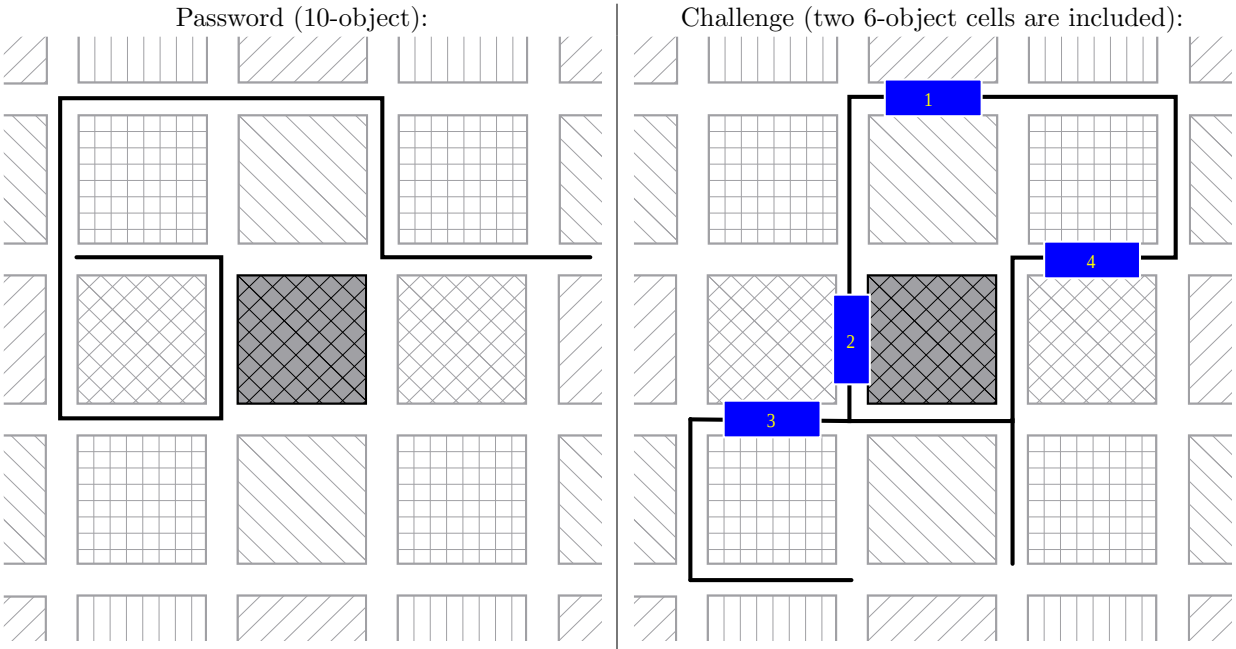


Figure 5: A snap of the online SecHCI system at `http://www.hooklee.com/SecHCI`

crosses. This example can be modified to several different versions, if the definition of the objects pool is different. Two different definitions of the objects pool are: the set of all possible turning directions at all crosses, and the set of all possible ways from a block to its neighbor blocks. The second definition corresponds to the DAS graphical password proposed by [31]. This implementation can be incorporated into electronic games based on maps, such as racing games and role-playing games.

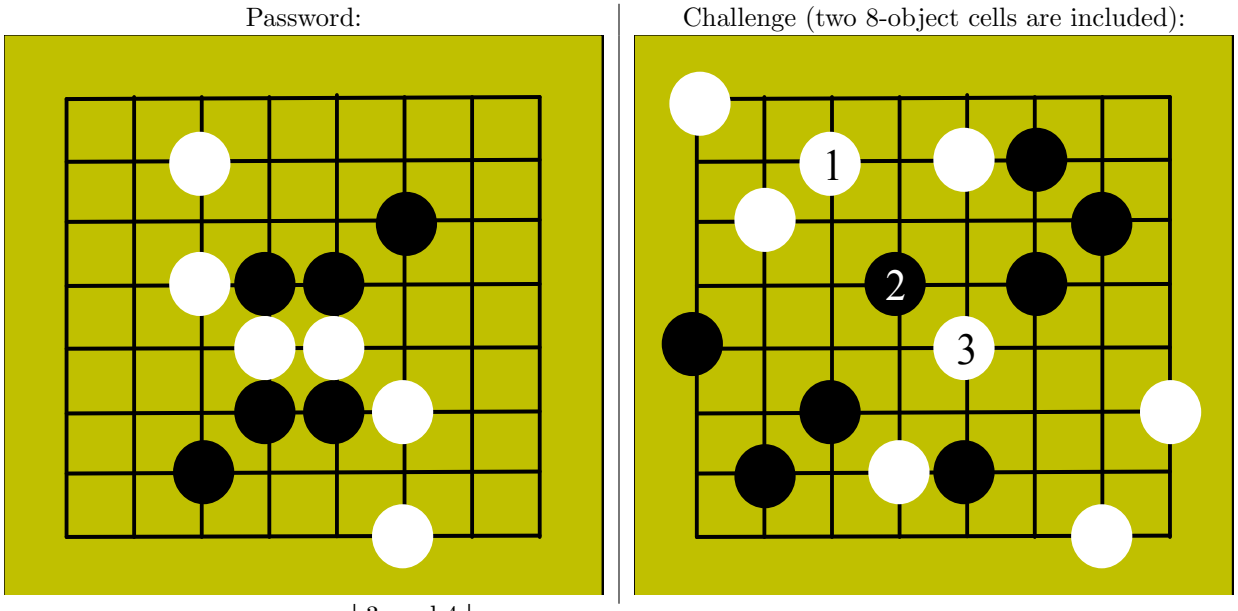
5.2.4 Yet Another Graphical Implementation Based on Chessboard

This implementation is based on a simplified chessboard of the game of I-go (see Figure 7 for one identification round). The objects pool is the set of all positions where the black and white chessmen are placed. In this implementation, only the fact that which positions are occupied is important, so the black chessmen and the white ones are equivalent. The use of two different kinds of chessmen is useful for users to remember passwords more easily and make responses more quickly. Also, if we use the color of chessmen as a part of the password, it may be possible to further enhance the security and usability of the involved Foxtail protocol. Naturally, this implementation can be easily extended to any chess-like game with a chessboard.



Response: $r = \left\lfloor \frac{4 \bmod 4}{2} \right\rfloor = 0$ (the blue blocks denote pass-objects)

Figure 6: A graphical implementation of Foxtail based on a traffic map: one identification round



Response: $r = \left\lfloor \frac{3 \bmod 4}{2} \right\rfloor = 1$ (the numbered chessmen denote pass-objects)

Figure 7: A graphical implementation of Foxtail based on a chessboard: one identification round

5.3 Security Analysis

5.3.1 Security against Brute Force Attacks

As a basic requirement, the password space $C(n,k)$ of this Foxtail protocol should be sufficiently large to resist brute force attacks. There are two kinds of brute force attacks: 1) offline attacks - once at least one identification is observed, one can exhaustively guess the password by checking the identification process; 2) online attacks - one can try to find the right password of an legal user by exhaustively guessing all possible ones. In some applications, such as ATM-s, online attacks are impossible since only several continuous trial failures are allowed. In all situations, offline attacks are always very dangerous when hidden peeping attacks run. To provide cryptographically strong security, $C(n,k) \geq 2^{80}$ is required [1]. In Table 1, we give some typical values of n, k, l and the corresponding levels of security for reference:

Table 1: Some reference parameters of the proposed Foxtail protocol

n	140	160	140	160	160	140	200
k	14	16	21	20	24	30	30
l	15	15	10	24	20	7	10
$C(n,k) \approx$	$2^{62.5}$	$2^{71.8}$	$2^{79.5}$	$2^{83.6}$	$2^{94.1}$	$2^{101.3}$	$2^{118.3}$

5.3.2 Balance Property

To make the response r balanced, $\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)$ should distribute uniformly in the set $\{0,1,2,3\}$. The simplest way to realize the uniform distribution is to generate both challenge-cells $\mathcal{C}_1, \mathcal{C}_2$ with **Uni-Rule**. However, only using **Uni-Rule** will make the protocol insecure to peeping attacks based on **partially-known passwords**: When a *passive* adversary $\mathcal{A}$ gets to know $k' \geq 3$ pass-objects in P, he can get some challenge-cells whose similarities are really 3 if all pass-objects in any challenge-cell are known by $\mathcal{A}$. Although such challenge-cells cannot directly tell $\mathcal{A}$ information of other pass-objects, they can reveal some elements not included in P. Then with $O(n)$ observations, $\mathcal{A}$ can get P (or an almost identical superset of P) by excluding revealed objects not in P with a high probability. If $\mathcal{A}$ does not know any partial information of P, he can exhaustively guess all 3-size subsets of $\mathcal{O}$ to try to get P with $O(n)$ observations. The total attack complexity is about $n \cdot C(n,3)$, which is much smaller than the complexity of simple brute force attacks $C(n,k)$.

To avoid the above partially-known password attack, one of $\mathcal{C}_1, \mathcal{C}_2$ must be generated at random in all possible challenge-cells with **Ran-Rule**. From the following Proposition 1, **Ran-Rule** together with **Uni-Rule** can guarantee the uniformity of $\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)$ in {0,1,2,3}. In fact, it is the reason why we introduce two different generation rules to generate two challenges for each challenge.

Proposition 1 *Assume a random variant x distributes in $A = \{0,1,2,\cdots,N-1\}$ $(N > 1)$ uniformly, and a random variant y does in A with **any** distribution series, it is true that $(x+y) \bmod N$ also distributes in A uniformly.*

Proof: $\forall i \in A$, we have

$$Pr[x + y \equiv i \pmod N] = \sum_{j=0}^{N-1} \left(Pr[x = j] \cdot Pr[y \equiv (i-j) \bmod N] \right).$$

Because $\forall j \in A$, $Pr[x = j] = \dfrac{1}{N}$,

$$\begin{aligned} Pr[(x+y) \bmod N = i] &= \sum_{j=0}^{N-1} \left(\frac{1}{N} \cdot Pr[y \equiv (i-j) \bmod N] \right) \\ &= \frac{1}{N} \cdot \sum_{k=0}^{N-1} Pr[y \bmod N = k] = \frac{1}{N}. \end{aligned}$$

That is to say, $(x+y)$ mod N distributes in A uniformly. The proof is complete. ■

Assume $\mathcal{C}_1$ is generated with **Uni-Rule**, $\mathrm{Sim}_P(\mathcal{C}_1)$ will distribute uniformly in $\{0,1,2,3\}$. From Proposition 1, it is obviously true that $\mathrm{Sim}_P(\mathcal{C}_1)+\mathrm{Sim}_P(\mathcal{C}_2)$ also distribute uniformly in $\{0,1,2,3\}$.

Besides the uniform distribution of $\mathrm{Sim}_P(\mathcal{C}_1)+\mathrm{Sim}_P(\mathcal{C}_2)$, for the challenge-cell generated with **Uni-Rule**, the occurrence probability of pass-objects and the probability of other objects not contained in P should also be balanced. When $\mathrm{Sim}_P(\mathcal{C})=0$, the occurrence probability of each pass-object is $\frac{0}{k}$; when $\mathrm{Sim}_P(\mathcal{C})\in\{1,2,3\}$, the probability is $\frac{C(k-1,\mathrm{Sim}_P(\mathcal{C})-1)}{C(k,\mathrm{Sim}_P(\mathcal{C}))}=\frac{\mathrm{Sim}_P(\mathcal{C})}{k}$. Thus, we can get the final occurrence probability of each pass-object:

$$\sum_{\mathrm{Sim}_P(\mathcal{C})=0}^{3}\frac{\mathrm{Sim}_P(\mathcal{C})}{k}\cdot\frac{1}{4}=\frac{1.5}{k}.$$

From the balance property, $\frac{1.5}{k}$ should be equal to the natural probability of any object occurring in each cell $\frac{C(n-1,l-1)}{C(n,k)}=\frac{l}{n}$. So we can get $\frac{1.5}{k}=\frac{l}{n}\Rightarrow 3n=2kl$.

5.3.3 Security against Passive Peeping Attacks

To measure the security against peeping attacks, we should firstly investigate how an adversary $\mathcal{A}$ get the equation system Eq. (1). To get the equation system, $\mathcal{A}$ can rewrite $\mathcal{O},P,\mathcal{C}_1,\mathcal{C}_2$ as n-length binary vectors $\boldsymbol{\mathcal{O}},\boldsymbol{P},\boldsymbol{\mathcal{C}_1},\boldsymbol{\mathcal{C}_2}$: if the i^{th} object is contained in the object set, then the i^{th} elements is 1, otherwise the i^{th} elements is 0. Thus, $\boldsymbol{\mathcal{O}}=(\overbrace{1,\cdots,1}^{n})$, and the similarity is the dot product of $\boldsymbol{\mathcal{C}}$ and $\boldsymbol{P}$: $\mathrm{Sim}_P(\mathcal{C})=\boldsymbol{\mathcal{C}}\cdot\boldsymbol{P}$.

In passive peeping attacks, for each observed challenge-response pair, $\mathcal{A}$ will get a equation

$$r=\left\lfloor\frac{(\boldsymbol{\mathcal{C}}_1\cdot\boldsymbol{P}+\boldsymbol{\mathcal{C}}_2\cdot\boldsymbol{P})\bmod 4}{2}\right\rfloor.$$

By guessing the right value of the hidden response $r_h=(\boldsymbol{\mathcal{C}}_1\cdot\boldsymbol{P}+\boldsymbol{\mathcal{C}}_2\cdot\boldsymbol{P})$ mod 4 from r with a probability equal to $\frac{1}{2}$, he can get a probabilistic equation $(\boldsymbol{\mathcal{C}}_1\cdot\boldsymbol{P}+\boldsymbol{\mathcal{C}}_2\cdot\boldsymbol{P})\equiv r_h \pmod 4$. With n equations independent[11] over GF(4), it is possible for $\mathcal{A}$ to uniquely solve the equations system (1) to get the password P. The probability to guess all right values of n hidden responses is 2^{-n}, that is to say, the proposed Foxtail protocol is $(2^{-n},O(n))$-secure against passive peeping attacks.

Here, please note that not all Foxtail maps can be used in the proposed protocol. Here, we give an example of bad Foxtail maps. Let us re-define the Foxtail map as $F(r_h)=r_h \bmod 2$. With such a Foxtail map, $F(f(c,P))$ are collapsed to be $(\mathrm{Sim}_P(\mathcal{C}_1)+\mathrm{Sim}_P(\mathcal{C}_2))$ mod 2. It is obvious that such a re-defined Foxtail protocol is not secure against passive peeping attacks, since the introduced uncertainty is removed and $\mathcal{A}$ can get an unambiguous linear equation $(\boldsymbol{\mathcal{C}}_1\cdot\boldsymbol{P}+\boldsymbol{\mathcal{C}}_2\cdot\boldsymbol{P})\equiv r \pmod 2$ for each challenge-response pair. As a result, once $\mathcal{A}$ get n equations independent over GF(2), he can directly get the password P. Then the modified Foxtail protocol becomes $(1,O(n))$-secure (i.e., insecure) against passive peeping attacks. So, we have a principle on the selection of the Foxtail map: **the introduced uncertainty cannot be removed by the composition of the Foxtail map $F(\cdot)$ and the function $f(c,P)$.**

5.3.4 Security against Active Peeping Attacks

The random permutation of all challenge objects makes it impossible for adversaries to distinguish which challenge-cell is generated with **Ran-Rule** and which one is generated with **Uni-Rule**. In addition, adversaries have no information on P, so adversaries can only passively replay challenge-cells generated with **Ran-Rule** in active peeping attacks. Under such a condition, maybe it is still possible for adversaries to get some useful information from **intersection attacks**, in which adversaries replay challenges with the least difference to users and try to find some

[11] Please note that the term independence here means the n equations are independent over the finite field GF(4), in which the addition of two elements a and b is defined as $(a+b)$ mod 4.

useful information to lessen attack complexity. Let us study what adversaries can get from such attacks. Assume the two challenges are $c = \{\mathcal{C}_1, \mathcal{C}_2\}$, $c' = \{\mathcal{C}'_1, \mathcal{C}'_2\}$, only one object is different for c and c': $o \in c$ and $o' \in c'$, where $o \neq o'$. The observed responses corresponding to the two challenges are r, r'. Assume $s = (\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)) \bmod 4$ and $s' = (\mathrm{Sim}_P(\mathcal{C}'_1) + \mathrm{Sim}_P(\mathcal{C}'_2)) \bmod 4$, Table 2 lists the possible values of s and s' for different relations of r, r' and o, o'.

Table 2: Possible values of (s, s') in intersection attacks

$r = r'$?	$o \in P$?	$o' \in P$?	(s, s')
Yes	Yes	Yes	(1,1), (2,2) or (3,3)
	No	No	(0,0), (1,1), (2,2) or (3,3)
	Yes	No	(1,0) or (3,2)
	No	Yes	(0,1) or (2,3)
No	Yes	No	(0,3) or (2,1)
	No	Yes	(1,2) or (3,0)

Apparently, adversaries can benefit nothing from intersection attacks. This fact implies that the proposed Foxtail protocol is also $(2^{-n}, O(n))$-secure against active peeping attacks.

5.3.5 Human Sensitivity to Active Peeping Attacks

In the above sub-subsection, we know that active adversaries can only use **Ran-Rule** to replay all challenges, i.e. it is possible for users to detect active adversaries via the difference between challenge-cells generated with **Uni-Rule**+**Ran-Rule** and those generated with **Ran-Rule**+**Ran-Rule**. That is to say, human sensitivity to active peeping attacks becomes possible. To investigate the difference between legal challenges and faked challenges, let us consider the distribution series of $\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)$ in two conditions respectively. We use $\mathcal{C}_{\mathbf{Ran}}$ to denote the challenge-cell generated with **Ran-Rule** and $\mathcal{C}_{\mathbf{Uni}}$ to denote the challenge-cell generated with **Uni-Rule**:

- Legal challenges:

$$\begin{aligned} Pr[\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2) = i] &= \sum_{j=0}^{\min(i,3)} Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Uni}}) = j] \\ &\quad \cdot Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = i - j] \\ &= \frac{1}{4} \cdot \sum_{j=0}^{\min(i,3)} Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = i - j] \end{aligned} \tag{2}$$

- Faked challenges:

$$\begin{aligned} Pr[\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2) = i] &= \sum_{j=0}^{\min(i,k,l)} Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = j] \\ &\quad \cdot Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = i - j] \end{aligned} \tag{3}$$

In the above equations, $Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = i] = \dfrac{C(k,i)C(n-k,l-i)}{C(n,l)}$. When $n = 140, k = 14, l = 15$, we can calculate the distribution series of both legal challenges and fake challenges, which are shown in Figure 8. Although there really exist difference between the two distribution series, the probability differences are too small (< 0.05) for humans to consciously notice the existence of active peeping attacks. The only event humans can absolutely detect active attacks is $\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2) > l + 3$, since in legal challenges the maximum of $\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)$ is $l + 3$. In a t-round Foxtail protocol, the probability that the above event occurs at least for one time, i.e. the probability that users can absolutely detect active attacks within one identification procedure, is

$$1 - (1 - Pr[\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2) > l + 3])^t. \tag{4}$$

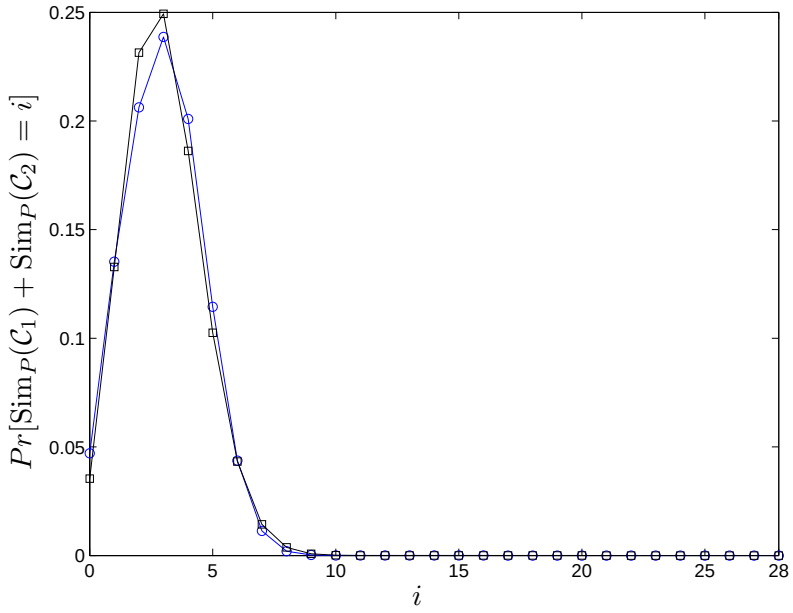


Figure 8: The distribution difference between legal challenges and fake challenges
(Legend: ○: legal challenges; □: fake challenges)

However, the above probability is so small that the proposed Foxtail protocol is almost $(1, k)$-human sensitive to active peeping attacks, that is to say, it is almost not human sensitive. When $n = 140, k = 14, l = 15, t = 20$, this probability is only about $1.756372824956998 \times 10^{-12}$.

To enhance human sensitivity to active peeping attacks, we have to make some modifications on the original protocol. One available method is to simplify the random permutation of all $2l$ challenge-objects in each challenge to the random permutation of the two l-object challenge-cells. With such a modification, once both challenge-cells in any challenge contain more than 3 objects, users can absolutely detect the existence of active peeping attacks. The corresponding probability becomes

$$1 - \left(1 - \sum_{i=4}^{\min(k,l)} \sum_{j=4}^{\min(k,l)} Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = i] \cdot Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = j]\right)^t . \tag{5}$$

When $n = 140, k = 14, l = 15, t = 20$, it is about $0.04 = \frac{1}{25}$, which means the modified Foxtail protocol is 0.04-human sensitivity to active peeping attacks. Apparently, the human sensitivity is comparatively weak (Hopper-Blum Protocol 2 is 0.1-human sensitive [30]), but is much better than the original protocol and sufficient in some applications.

The lack of strong human sensitivity is a major weakness of the Foxtail protocol proposed in this paper. How to increase the human sensitivity is an emphasized issue in our future research on SecHCI.

5.3.6 Security against Virtual Partial Passwords Based Attacks (VPPA)

In Sec. 5.3.2, we have mention a kind of attacks based on partially-known passwords and show the importance of **Uni-Rule** to resist such attacks. In this sub-subsection, we will focus another kind of attacks based on partially-known passwords.

When an adversary $\mathcal{A}$ gets to know $k' \geq 3$ pass-objects, if $\mathcal{A}$ use the k' pass-objects as a **virtual (partial) password** $P' \subset P$ to make all responses, it can be expected that the success probability may be greater than 2^{-t}. If the probability is sufficiently large, it is possible for $\mathcal{A}$ to pass the identification by accident after a small number of trial failures. Actually, even if $\mathcal{A}$ does not know anything about the password P, he can exhaustively try

all possible k'-size passwords to find a valid virtual password P'. Since the virtual password space, i.e. the total number of k'-size passwords $C(n,k')$, is generally much smaller than the password space $C(n,k)$, so the exhaustive test may be feasible in practice. Actually, in such attacks, time complexity (the average time to reach one success identification with **virtual responses**) is borrowed to effectively reduce the space complexity (the password space). If $\mathcal{A}$ can use many hosts on a large-scale network to carry out distributed attacks, the space complexity can be further reduced.

Now let us study what the success probability will be in VPPA. Firstly, to facilitate the following discussion, define the **virtual similarity** of a challenge-cell $\mathcal{C}$ as $\mathrm{Sim}'_P(\mathcal{C}) = \#(\mathcal{C} \cap P')$, and define three probabilities $P_{a'|a}$, $P_a^{\mathbf{Ran}}$ and $P_a^{\mathbf{Uni}}$ as follows:

1) $\forall a = 0 \sim \min(k,l)$ and $a' = 0 \sim \min(k',a)$:

$$P_{a'|a} = Pr[\mathrm{Sim}'_P(\mathcal{C}_{\mathbf{Ran}}) = a' | \mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = a] = \frac{C(k',a')C(k-k',a-a')}{C(k,a)}; \tag{6}$$

2) $\forall a = 0 \sim \min(k,l)$:

$$P_a^{\mathbf{Ran}} = Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Ran}}) = a] = \frac{C(n-k,l-a)C(k,a)}{C(n,l)}; \tag{7}$$

3) When $a = 0,1,2,3$:

$$P_a^{\mathbf{Uni}} = Pr[\mathrm{Sim}_P(\mathcal{C}_{\mathbf{Uni}}) = a] = \frac{1}{4}. \tag{8}$$

Then we can calculate the success probability of VPPA in one identification round:

$$P_{\mathrm{VPPA}} = \sum_{(a',a,b',b)} B_{(a',a,b',b)} \cdot P_{a'|a} \cdot P_a^{\mathbf{Ran}} \cdot P_{b'|b} \cdot P_b^{\mathbf{Uni}}, \tag{9}$$

where

$$B_{(a',a,b',b)} = \begin{cases} 1, & \left\lfloor \frac{a'+b' \bmod 4}{2} \right\rfloor = \left\lfloor \frac{a+b \bmod 4}{2} \right\rfloor, \\ 0, & \left\lfloor \frac{a'+b' \bmod 4}{2} \right\rfloor \neq \left\lfloor \frac{a+b \bmod 4}{2} \right\rfloor, \end{cases} \tag{10}$$

and $a, a' = 0 \sim \min(k',l)$, $b, b' = 0 \sim 3$. When $n = 140, k = 14, l = 15$ (typical values of the proposed Foxtail protocol), the value of P_{VPPA} with respect to k' is shown in Figure 9. We can see when $k' < 8$ the probability is less than 0.5 (the success probability of random response), so VPPA is even worse than normal attacks. When $k' \geq 8$, P_{VPPA} is greater than 0.5, but it is expected that VPPA cannot work efficiently as a tool to break the proposed Foxtail protocol since $C(n,8) \approx 2^{41.44}$. For other values of n,k,l, similar results are obtained to support this conclusion.

5.3.7 Security against Administrator Attacks

In identification systems based on fixed passwords, generally the verifiers do not know the passwords, but their hash values [1]. This fact is useful to avoid the danger that administrators get your password and steal your identity. However, in the proposed Foxtail protocol, legal users share their secret passwords with the verifier. So your secrets are open to dishonest administrators. How to avoid such dangers? There exist a possible way: to use (local) secret objects with the shared passwords. With such a method, the password known by administrators becomes meaningless if administrators do not know the relations between the passwords and the (local) secret objects.

As an example, for the icon-based SecHCI system introduced in Sec. 5.2.2, the n icons can be locally saved on your computer, and you set your password in your computer and send the meaningless pass-bits to the remote server for storage. To generate the challenges, a (local) client program should be used to translate the pseudo-random numbers (sent from the remote server) representing $2l$ challenge-objects to appropriate (local) icons displayed in the (local) screen.

If an administrator wants to steal a user's identity, he has to get the control of the user's private computer, which is impossible before he successfully steals the user's identity. Apparently, he meets a contradiction and the SecHCI system becomes secure against administrator attacks. Furthermore, even if the administrator has successfully stolen

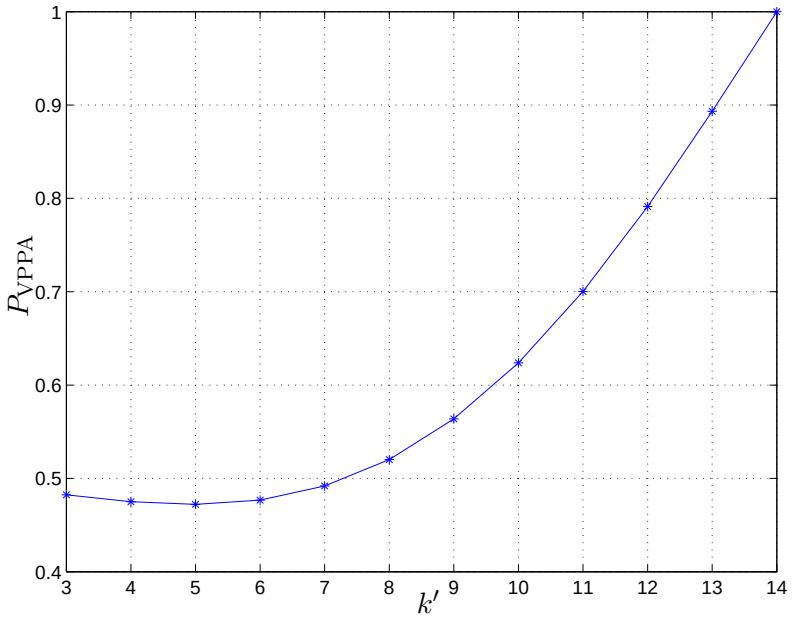


Figure 9: The value of P_{VPPA} with respect to k', where $n = 140, k = 14, l = 15$.

your local objects, it still possible to make him impossible to login as your identity, because he does not know the relationship between the objects and your pass-bits saved in the remote server. The relationship is controlled by the local translation program, and the only way to get the relationship is to crack this local program. However, it is still impossible before an administrator successfully steal your identity. The administrator meets another contradiction again.

Actually, legal users can only remember k pass-objects, and the pass-bits saved in the remote server are entirely meaningless for them. That is to say, the use of local objects can effectively insulate (private) users' passwords from publicly-saved passwords. In Sec. 5.4.3, we will continue to discuss the influence of local objects on the usability of the proposed SecHCI protocols.

5.4 Usability Analysis

Many different factors are involved in the usability of a SecHCI protocol: what the password is and how long it is; how difficult the identification procedure is[12]; how long the identification time is; how much memory is needed to store the objects pool and passwords of all users; how much the display space of each challenge is in the screen; and so on. These factors reflect limited capabilities of both humans and computers: humans cannot remember long passwords; humans will be tired to make responses for a long time; humans cannot make complicate computations quickly and exactly; computers cannot store too many objects in a limited memory; computer screens cannot display too many objects simultaneously, etc. The list will become even longer for disabled humans and digital devices with more limited capabilities. To design a SecHCI protocol with good usability, all involved issues should be considered carefully.

Basically speaking, the usability of the proposed Foxtail protocol in this section is not sufficiently good, but we believe SecHCI protocols with better security and usability can be found in future following the design route described in this paper. All discussions given in this subsection can be qualitatively extended to other SecHCI protocols.

[12]If the Foxtail map is too complicated in mathematics, maybe kids cannot calculate the right responses at all. Please see Sec. 6.2 for applications of this factor.

5.4.1 A Basic Evaluation

What the password is The password contains k different pass-objects, which are taken from n objects. In textual implementations, the password corresponds to traditional passwords widely-used in most identification systems; and in graphical implementations, the password corresponds to the visual/graphical passwords introduced in Sec. 4.2.3. Basically speaking, as many researchers have pointed out in literature on visual passwords, both the security and the usability of graphical implementations are better than textual implementations (at least a better trade-off can be achieved).

How long the password is The password length is k, which is fixed in the proposed Foxtail protocol. To enhance the usability, smaller k is desired. In the next sub-subsection, we will introduce a typical way to help users to remember long passwords more easily, which is more useful for graphical implementations of the proposed Foxtail protocol.

How difficult the identification procedure is The computations involved in identification procedure include: addition, $\bmod 4$ operation, division, and $\lfloor\cdot\rfloor$ operation. In fact, because only $0, 1, 2, 3$ are concerned in division, and $\lfloor\cdot\rfloor$ operation, most humans will not do the two operations, but try to tell which set $(\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)) \bmod 4$ lies in: $\{0, 1\}$ or $\{2, 3\}$. The most difficult operation is $\bmod 4$, and almost all humans can do it after simple training.

How long the identification time is The consuming time for each identification is $\tau_0 \times t$, where τ_0 is the mean response time for one challenge. Generally speaking, the larger n, k, l are, the longer the time will be. Our experimental studies show that graphical implementations can reduce the identification time dramatically (less than half) in comparison to textual implementations.

How much memory is needed to store the objects pool and passwords of all users The objects pool contains n objects and each password contains k objects. Since n and k are not very large, all computers and almost all PDA-s can store the objects pool and a number of passwords in their memory. However, if the number of users is too many, the required memory may be too large. As a result, to save memory and support more registered identities, the textual implementations are better than graphical implementations. Of course, if the verifier is a remote server with sufficient memory, this problem can become trivial.

How much the display space of each challenge is in the screen In each challenge, at least $2l$ objects should be displayed in the screen. Considering the display screen of PDA-s is generally small, the less space $2l$ objects hold, the better the usability will be. Therefore, the smaller l is and the smaller the size of a single object is, the better the usability will be. Comparing the four different implementations given in Sec. 5.2, we can see the textual implementation requires the least display space, and the icon-based implementation requires the largest. Generally speaking, pictures with large sizes, such as those used in [4], should be avoided as possible in the proposed SecHCI protocol.

A paradox on implementation From the above discussion, we can see there exists a paradox in the implementation of the proposed SecHCI protocol: textual implementations are better to save the display space and the required memory of local client devices (computers or PDA-s) and remote servers, but graphical implementations are better for other aspects of the usability, and it has been shown [4, 31, 50, 53] that graphical passwords have better capability to resist dictionary attacks [1, 62] than textual passwords. As a compromise, the grid-based graphical implementations, such as the last two ones introduced in Sec. 5.2.3 and 5.2.4, may be useful to achieve a better overall usability. Also, graphical implementations based on small icons are also accepted.

Yet another paradox In addition, there exists another paradox between security and usability. Better security wants n, k, l larger, but better usability wants n, k, l smaller. As a natural result, the security and usability must be balanced in consideration of requirements in different applications. The parameters $n = 140, k = 14, l = 15$ can be considered as a reference of an acceptable configuration for today's computer technology.

5.4.2 Pass-Rule: Remember Your Password More Easily

Remembering passwords is often a very annoying problem for many users in the security world. The use of multiple passwords and the security policy forcing users to change passwords frequently make this problem even worse. In many cases, users have to discard their needs for security to approach convenience, which makes any cryptosystem fail [47, 48].

The main interest of using visual/graphical passwords is to relax this problem about traditional passwords (see Sec. 4.2.3). Unfortunately, following psychology research [50, 63, 64], it is still rather hard for humans to exactly remember more than 8 pass-pictures and fluently make all responses without errors. To assist humans to remember their passwords more easily, we suggest using a secret **pass-rule** (which is similar to pass-algorithm proposed in [33]) to remember many pass-objects.

Basically, a **pass-rule** is a specific ***common*** feature of all pass-objects, which can help users to distinguish whether or not a given object is a pass-object. Actually, we can even forget our passwords, but remember only the corresponding pass-rules as equivalent passwords. Since humans are strong to remember meaningful things, it will be much easier for humans to remember pass-rules than many different pass-objects.

For example, in the icon-based implementation of the proposed Foxtail protocol, you can set your pass-rule as "*symmetric national flags*", which is an easy pass-rule to remember k different pass-icons. Of course, it is your duty to ensure all pass-icons yield to this pass-rule and all others do not. To do so, you have to check all n icons and change some icons to others. Apparently, if the objects pool $\mathcal{O}$ is fixed and cannot be customized by users, it will be more difficult to find pass-rules for their passwords. Therefore, for a good SecHCI system, it is an important feature to allow users to freely re-define the objects pool.

If the objects pool can be freely customized, it is expected that everybody can easily find a good pass-rule that is simple enough for himself but impossible for all others (of course, also impossible for robots to run automatic dictionary attack). A common way to select a practically strong pass-rule is to combine several simple sub-rules: you can select your pass-rule as "national flags with three major colors and at least one stars", which will be much more difficult for others to guess. Another way is to use several independent pass-icons that are not classified into the major pass-rule, such as 12 "faces" (the major pass-rule) plus "flag of my own nation and a handgun" (independent pass-icons). For more discussion on pass-rule, please refer to Sec. 4.4.2 of [40].

5.4.3 Using Local and Removable Objects

In Sec. 5.3.7, we have discussed the use of local objects to enhance security of the proposed SecHCI protocol against administrator attacks. In addition, using local objects is also helpful for users to find/remember a pass-rule more easily and make responses more fluently. It is natural since users are more familiar to files saved in their own computers.

Another merit of using local objects is about the network traffic. In picture-based graphical implementations, when the pictures corresponding to pass-objects are all stored in remote server, the files corresponding to challenge-objects should be transmitted over network from the server to the client for display, which may exhaust the network traffic when the size of challenge-objects and/or the number of simultaneous users increase too much. With the use of local objects, the transmission of the pictures are cancelled, and only the bits representing challenges can be transmitted. Thus, the network traffic will be dramatically reduced and the display speed of each challenge can be promoted.

In mobile environments, it is natural to make the "local" objects removable by saving them on removable media, not on hard disks of local computers. Removable objects can be used as private tokens to enhance the security and usability of SecHCI. As we mentioned above, such tokens are different from traditional secure tokens, they are not sensitive to theft and loss because no secret information is saved on the removable media. Considering the prevalence of USB-disks[13] in recent years, it becomes easy to save hundreds and even thousands objects in mobile media. For example, in the icon-based graphical SecHCI system introduced in Sec. 5.2.2, the size of $n = 140$ icons is only 179,725 bytes, which is sufficiently small for any mobile media (even for outdated floppy disks).

[13] USB-disks generally have many mega-bytes storage capacity and are provided in computer market with acceptable prices. It is expected that USB-disks will play important roles in future digital world.

5.4.4 Usability Issues on the Handicapped

A good SecHCI system should be extended to support the special needs of the handicapped. Here, we discuss how to design the implementation of the proposed SecHCI protocol to facilitate the color blind and the blind.

The color blind For the color blind, the customization of the objects pool becomes more important than normal men, which makes it possible for the color blind men to determine the objects pool and their passwords to work well with their limited visual capability. Of course, some specially-designed tools can be developed to further help them to make choices.

The blind For the handicapped users who lose their visions, audio interface should be used instead of visual one to tell the users what the current challenge is. Also, printed cards with brailles may be another candidate to assist them. For users who lose both visual and audio capability, the latter is the only available method. Of course, special printers are needed to make braille cards, so the cost of the SecHCI system will be increased.

6 Some Special Topics on SecHCI

6.1 Eye-Tracking: A Powerful Peeping Technology in Future

In recent years, eye-tracking technique has been developed to enhance the performance of human-computer interface [65, 66]. When mini eye-tracking devices are used in peeping attacks, even for the proposed SecHCI protocol in this paper, it is possible for attackers to get some information about passwords, because we always unconsciously gaze the pass-objects for more time than other objects. Although the present eye-tracking technique is not so accurate and the size of eye-tracking devices is too large, considering the rapid advance in this area, it can be expected to be one of the most powerful tool to threaten our passwords in near future.

To frustrate eye-tracking based attacks, we suggest users should consciously control their eye-gazing behaviors to avoid noticeable attention on pass-objects. Limiting the display space of each challenge is also helpful to resist such peeping attacks, because at present eye-tracking devices are not sufficiently good to distinguish small changes in eye-movements. Recall the four example implementations given in Sec. 5.2, we can see the last two ones may have higher security against eye-tracking based attacks, since challenges can be constrained within a very small grid[14].

6.2 Protect Kids from Digital Pornography via SecHCI

Age Verification Technologies (AVT) have been used in practice to protect kids from online pornographic materials [67]. Here, we discuss the possibility of using SecHCI as a new AVT tool.

Apparently, if a SecHCI system is sufficiently hard so that only adults can perform the identification successfully, then it will be available as an age verification system to determine whether or not a user is an adult and the required pornographic materials should be displayed. We can see it is a fresh and interesting application of SecHCI. Of course, in such a system, since kids have great simulating capability, the function against peeping attack is still needed to prevent kids from mimicking their parents to pass the age verification procedure. Here, ***the usability can be relaxed***, since it is adults' responsibility to protect their kids when they enjoy pornography materials.

Here, we can give an modified version of the proposed Foxtail protocol to demonstrate how to make the identification procedure difficult only for kids. A positive integer $d \in \{2, \cdots, 7, 8\}$ is randomly generated together with each challenge and both are shown to users. The correct response should be yield to the following equation:

$$r = \left\lfloor \frac{((\mathrm{Sim}_P(\mathcal{C}_1) + \mathrm{Sim}_P(\mathcal{C}_2)) - d \times \max(\mathrm{Sim}_P(\mathcal{C}_1), \mathrm{Sim}_P(\mathcal{C}_2))) \bmod 4}{2} \right\rfloor . \tag{11}$$

Apparently, pupils who have no idea of multiplication and/or negative integers cannot make responses correctly, and it is either not easy to teach them to do so. Although the usability of such a modified Foxtail protocol becomes

[14] In fact, the distance between adjacent objects can be only two pixels, which is too small even for future eye-tracking devices tens years later.

worse, it is worth doing so to protect kids from improper contents (especially pornographic materials). For adults who can execute the above SecHCI, it is just a small problem to take more time to login before they get what they want.

What's more, the above idea can be further extended to verify children at different ages, and then it is possible to allow different children to access different materials suitable for their ages. To do so, multiple algorithms with different difficult levels should be used simultaneously.

7 Conclusion

In this paper, we address the dangers of peeping attacks in the real world and introduce Secure Human-Computer Identification system (SecHCI) against such attacks. After a brief survey on SecHCI and introduce formal definitions on peeping attacks and SecHCI, we propose some principles and two general structures to design SecHCI systems: Twins and Foxtail. To show the feasibility of the proposed principles, a concrete Foxtail protocol is presented and its different implementations are discussed (an icon-based implementation has been developed at http://www.hooklee.com/SecHCI). The security and usability of the proposed Foxtail protocol are investigated in detail. Although the proposal still has some nontrivial problems on security and usability, it is expected that some better SecHCI systems will be developed following the design experiences given in this paper. We hope this paper can stir more further research on this subject and leads to the final success against peeping attacks.

Acknowledgement

The authors would like to thank Prof. Manuel Blum at Carnegie Mellon University for his valuable suggestions on this work.

References

[1] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, Inc., 1996. 1, 1, 1.2, 4.2.1, 4.2.3, 5.3.1, 5.3.7, 5.4.1

[2] Ross J. Anderson. Why cryptosystems fail. In *Proc. 1st ACM Conf. Computer and Communication Security (CCS'93)*, pages 215–227, 1993. 1.1, 1.2

[3] Ross J. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37(11):32–40, 1994. 1.1, 1.2

[4] Rachna Dhamija and Adrian Perrig. Déjà Vu: A user study using images for authentication. In *Proc. 9th USENIX Security Symposium*, pages 45–58, 2000. 1.1, 2.1, 2.2, 4.1.2, 1, 5.2.2, 5.4.1, 5.4.1

[5] searchSecurity.com. Shoulder surfing. Available online at http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci802244,00.html, 14 February 2002. 1.1

[6] BBC News. Chinese cameras spying on spouses. Available at http://news.bbc.co.uk/hi/english/world/asia-pacific/newsid_1885000/1885218.stm, 21 March 2002. 1.1

[7] Mark Mitechell. Always on the lookout. *TIME Asia Maganize*, 159(12):15–27, April 2002. Available at http://www.time.com/time/asia/magazine/article/0,13673,501020401-219895,00.html. 1.1

[8] Cassi Goodman. An introduction to TEMPEST. Available online at http://rr.sans.org/encryption/TEMPEST.php, 18 April 2001. 2

[9] David O. Tyson. Emissions from bank computer systems make eavesdropping easy, expert says. American Banker, page 1, 22, available online at http://www.politrix.org/foia/nsa/nsa-vaneck.htm, March 26 1985. 1.1

[10] Barry Fox. New-wave spies: Electronic eavesdropping is becoming mere child's play. *New Scientist*, 164(2211):11, 6 November 1999. 1.1

[11] Joe Loughry and David A. Umphress. Information leakage from optical emanations. *ACM Trans. Information and System Security,*, 5(3):262–289, 2002. 1.1, 2.1

[12] Markus G. Kuhn. Optical time-domain eavesdropping risks of CRT displays. In *Proc. 2002 IEEE Sym. Security and Privacy (S&P'02)*, pages 1–16. IEEE Computer Society, 2002. 1.1, 2.1

[13] Frank Jones. Nowhere to run... nowhere to hide...: The vulnerability of CRT's, CPU's and peripherals to TEMPEST monitoring in the real world. Available online at http://www.parallaxresearch.com/dataclips/pub/infosec/emsec_tempest/info/TEMPESTMonitor.txt, 1996. 1.1

[14] HSBC. HSBC statement on fraudulent websites. Available online at http://www.hsbc.com.hk/hk/aboutus/press/content/03dec05e.htm, 5 December, 2003. 1.2

[15] Wikipedia. Phishing. online document available at http://en.wikipedia.org/wiki/Phishing, 2005. 1.2

[16] Aviel D. Rubin. Independent one-time passwords. *Computing Systems*, 9(1):15–27, 1996. 1.2

[17] Neil Haller. The S/Key$^{\text{TM}}$ one-time password system (also known as Internet RFC 1760). In *Proc. 1994 Symposium on Network and Distributed Systems Security (NDSS'94)*, pages 151–157. IEEE Computer Society, 1994. 1.2

[18] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO'86*, Lecture Notes in Computer Science **263**, pages 186–194. Springer-Verlag, Berlin, 1987. 1.2

[19] Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In *Proc. the 9th annual ACM conference on Theory of computing (STOC'87)*, pages 210–217. ACM Press New York, 1987. 1.2

[20] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology - EUROCRYPT'88*, Lecture Notes in Computer Science **330**, pages 123–128. Springer-Verlag, Berlin, 1988. 1.2

[21] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO'89*, Lecture Notes in Computer Science **435**, pages 239–252. Springer-Verlag, Berlin, 1990. 1.2

[22] Moni Naor and Benny Pinkas. Visual authentication and identification. In *Advances in Cryptology - CRYPTO'97*, Lecture Notes in Computer Science **1294**, pages 322–336. Springer-Verlag, Berlin, 1997. 1.2

[23] Kazukumi Kobara and Hideki Imai. Limiting the visible space visual secret sharing schemes and their application to human identification. In *Advances in Cryptology - ASIACRYPT'96*, Lecture Notes in Computer Science **1163**, pages 185–195. Springer-Verlag, Berlin, 1996. 1.2, 2.1

[24] Jay Stanley and Barry Steinhardt. Drawing a blank: The failure of facial recognition technology in Tampa, Florida. An ACLU Special Report, Available online at http://www.aclu.org/issues/privacy/drawing_blank.pdf, January 2002. 1.2

[25] Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, and Satoshi Hoshino. Impact of artificial "gummy" fingers on fingerprint systems. In *Optical Security and Counterfeit Deterrence Techniques IV*, Proceedings of SPIE vol. 4677, pages 275–289. SPIE–The International Society for Optical Engineering, 2002. 1.2

[26] ACLU. Airport security: Increased safety need not come at the expense of civil liberties. Available online at http://www.aclu.org/safeandfree/facts-airport.html, 2002. 1.2

[27] Manuel Blum and Nick Hopper. Cs 827: Security and cryptography. Available online http://www-2.cs.cmu.edu/~hopper/cs827-f01, September 2001. 2.1, 4, 2.2

[28] Aladdin Center of Carnegie Mellon University. The CMU HumanAut project. Available at http://www.captcha.net/humanaut, 2002. 2.1, 4

[29] Aladdin Center of Carnegie Mellon University. Human interactive proofs (HIPs). Available at http://www.aladdin.cs.cmu.edu/hips, 2002. 4

[30] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *Advances in Cryptology - ASIACRYPT 2001*, Lecture Notes in Computer Science **2248**, pages 52–66. Springer-Verlag, Berlin, 2001. 1.2, 2.1, 2.2, 3, 3.3, 7, 4.1.2, 4.2.1, 4.2.1, 4.2.2, 4.3.1, 5.3.5

[31] Ian Jermyn, Alain Mayer, Fabian Monrose, Michael K. Reiter, and Aviel D. Rubin. The design and analysis of graphical passwords. In *Proc. 8th USENIX Security Symposium*, pages 1–14, 1999. 2.1, 4.2.3, 3, 5.2.3, 5.4.1

[32] National Security Agency. National Security Agency specification for shielded enclosures. Specification NSA No. 94-106, available at http://cryptome.org/nsa-94-106.htm, 24 October 1994. 2.1

[33] James A. Haskett. Pass-algorithms: A user validation scheme based on knowledge of secret algorithm. *Communications of the ACM*, 27(8):777–781, 1984. 2.1, 2.2, 5.4.2

[34] Sidney L. Smith. Authentication users by word association. *Computers & Security*, 6(6):464–470, 1987. 2.1, 2.2

[35] Tsutomu Matsumoto and Hideki Imai. Human identification through insecure channel. In *Advances in Cryptology - EUROCRYPT'91*, Lecture Notes in Computer Science **547**, pages 409–421. Springer-Verlag, Berlin, 1991. 2.1, 2.2, 4.1.2, 4.2.1, 4.2.1

[36] Chih-Hung Wang, Tzonelih Hwang, and Jiun-Jang Tsai. On the Matsumoto and Imai's human identification scheme. In *Advances in Cryptology - EUROCRYPT'95*, Lecture Notes in Computer Science **921**, pages 382–392. Springer-Verlag, Berlin, 1995. 2.1, 2.2

[37] Tsutomu Matsumoto. Cryptographic human identification. In *Analysis, Design and Evaluation in Human-Computer Interaction*, volume III of *Proc. 6th Int. Conf. on Human-Computer Interaction (HCI International'95)*, pages 147–152, 1995. 2.1, 2.2, 4.1.2

[38] Tsutomu Matsumoto. Human-computer cryptography: An attempt. In *Proc. ACM Conf. on Computer and Communication Security*, pages 68–75. ACM Press, 1996. 2.1, 2.2, 4.1.2

[39] Nicholas J. Hopper and Manuel Blum. A secure human-computer authentication scheme. Technical Report of Carnegie Mellon University CMU-CS-00-139, 2000. 2.1, 2.2, 4.2.1

[40] Shujun Li and Heung-Yeung Shum. Secure human-computer identification against peeping attacks (SecHCI): A survey. Unpublished technical report, 2002. A draft is available online at http://www.compscipreprints.com/comp/Preprint/hooklee/20030121/1/. 2.1, 2.2, 3, 3.1, 4.2.3, 4.2.3, 4.2.3, 5.4.2

[41] Nick Hopper. Security and complexity aspects of human interactive proofs. Presented at the First Workshop on Human Interactive Proofs (HIP), abstract available at http://www.aladdin.cs.cmu.edu/hips/events/abs/hopper_abstract.pdf, 2002. 2.2

[42] N. J. Hopper. Answer e-mails to "some questions about your paper on AsiaCrypt 2001". Private Communications between N. J. Hopper and Shujun Li, August 2002. 2.2

[43] Bruce Schneier. *Applied Cryptography – Protocols, algorithms, and souce code in C.* John Wiley & Sons, Inc., New York, second edition, 1996. 4.1.1

[44] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proc. 6th ACM Conf. Computer and Communications Security (CCS'99)*, pages 28–36, 1999. 4.2.1

[45] Carl Ellison, Chris Hall, Randy Milbert, and Bruce Schneier. Protecting secret keys with personal entropy. *Future Generation Computer Systems*, 16(4):311–318, 2000. 4.2.1

[46] Ari Juels and Martin Wattenberg. Error-tolerant password recovery. In *Proc. 8th ACM Conf. Computer and Communications Security (CCS'01)*, pages 1–9, 2001. 4.2.1

[47] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc., New York, 2000. 4.2.2, 5.4.2

[48] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):41–46, 1999. 4.2.2, 5.4.2

[49] Martina Angela Sasse, Sacha Brostoff, and Dirk Weirich. Transforming the 'weakest link' – a human/computer interaction approach to usable and effective security. *BT Technology J.*, 19(3):122–131, 2001. 4.2.2

[50] David Bensinger. Human memory and the graphical password. Available at http://www.passlogix.com/media/pdfs/bensinger.pdf, 1998. 4.2.3, 2, 5.4.1, 5.4.2

[51] Sacha Brostoff and M. Angela Sasse. Are Passfaces more usable than passwords? a field trial investigation. In *People and Computers XIV - Usability or Else (Proceedings of HCI 2000)*, pages 405–424, Sunderland, UK, 5–8, September 2000. Springer, Berlin. 1, 5.2.2

[52] Real User Corporation. PKI and Passfaces™: Synergistic or competitive? Available at http://www.realuser.com/published/PassfacesAndPKI.pdf, October 2001. 1, 5.2.2

[53] Real User Corporation. The science behind Passfaces. Available at http://www.realuser.com/published/ScienceBehindPassfaces.pdf, September 2001. 1, 5.2.2, 5.4.1

[54] Real User Corporation. Real User Corporate profile. Available at http://www.realuser.com/published/RealUserCorporateProfile.pdf, April 2002. 1, 5.2.2

[55] ID Arts Inc. Passfaces - the art of identification. Please visit http://www.idarts.com, 2002. 1, 5.2.2

[56] Tetsuji Takada and Hideki Koike. Awase-E: Image-based authentication for mobile phones using user's favorite images. In *Human-Computer Interaction with Mobile Devices and Services: 5th International Symposium Proceedings*, Lecture Notes in Computer Science **2795**, pages 347–351. Springer-Verlag, Berlin, 2003. 1, 5.2.2

[57] Vince Sorensen. PassPic - Visual password management. Please visist http://www.authord.com/PassPic, 2002. 2

[58] Marc Boroditsky. v-GO™: Usable security technology. Available at http://www.passlogix.com/media/pdfs/usable_security.pdf, 2000. 2

[59] Passlogix Inc. Welcome to passlogix. Please visit http://www.passlogix.com, 2002. 2

[60] Aladdin Center of Carnegie Mellon University. The CAPTCHA project. Available at http://www.captcha.net, 2002. 4.2.3

[61] Lius von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart (automatically) or How lazy cryptographers do AI. Technical Report CMU-CS-02-117, Carnegie Mellon University, Feb. 2002. 4.2.3

[62] Daniel V. Klein. "Foiling the cracker": A survey of, and improvements to, password security. In *Proc. 2nd USENIX Security Workshop*, pages 5–14, 1990. 5.4.1

[63] G. Miller. The magic number seven plus or minus two: Some limits on your capacity for processing information. *Psychological Review*, 63(1):81–96, 1956. 5.4.2

[64] Rachel Rue. Eighty-six bits of memory magic. Presented at the First Workshop on Human Interactive Proofs (HIP), abstract available at http://www.aladdin.cs.cmu.edu/hips/events/abs/rue_abstract.pdf, 2002. 5.4.2

[65] Robert J.K. Jacob. Eye tracking in advanced interface design. In W. Barøoeld and T. Furness, editors, *Advanced Interface Design and Virtual Environments*, pages 258–288. Oxford University Press, Oxford, 1995. 6.1

[66] Jingtao Wang, Shumin Zhai, and Hui Su. Chinese input with keyboard and eye-tracking - an anatomical study. In *Proceedings of the SIGCHI conference on Human factors in computing systems 2001*, pages 349–356, Seattle, Washington, United States, 2001. ACM Press. 6.1

[67] Dick Thornburgh and Herbert S. Lin, editors. *Youth, Pornography and the Internet*. National Acamedy Press, Washington, D.C., 2002. 6.2