Addressing Online Dictionary Attacks with Login Histories and Humans-in-the-Loop*

S. Stubblebine[1], P.C. van Oorschot[2]

[1] Stubblebine Research Labs, New Jersey, USA
[2] Computer Science, Carleton University, Canada

Abstract. Automated Turing Tests (ATTs), also known as human-in-the-loop techniques, were recently employed in a login protocol by Pinkas and Sander (2002) to protect against online password-guessing attacks. We begin by noting that this, and other protocols involving ATTs, are susceptible to minor variations of well-known middle-person attacks. We discuss techniques to address such attacks, and present complementary modifications in a new history-based protocol with ATTs. Analysis indicates that the new protocol offers opportunities for improved security and user-friendliness (fewer ATTs to legitimate users), and greater flexibility (e.g. allowing protocol parameter customization for particular situations and users).

1 Introduction

Interest has recently been rekindled in tests distinguishing humans from computers [21], and using these to ensure human involvement in a broad range of computer-based interactions. The idea is to find simple tasks relatively easily performed by humans, but apparently difficult or infeasible for automated programs – such as visually recognizing distorted words. Mechanisms involving such techniques have been called *mandatory human participation schemes*, *human-in-the-loop protocols*, and *Automated Turing Tests* (ATTs) [19, 5, 23].

Among others, one specific purpose for which such tests have been proposed is protecting web sites against access by automated scripts. ATTs are currently being used to protect against database queries to domain registries, to prevent sites from being indexed by search engines, and to prevent "bots" from signing up for enormous numbers of free email accounts [5]. They have also been proposed for preventing more creative attacks [4].

We are primarily interested in the use of ATTs to protect web servers against online password guessing attacks (e.g. online dictionary attacks). The idea is that automated attack programs will fail the ATT challenges. A specific instance of such a protocol was recently proposed by Pinkas and Sander [20]. While this protocol appears to be quite simple, closer inspection reveals it to be surprisingly subtle and well-crafted. Simpler techniques preventing online dictionary attacks are not always applicable. For example, account lock-out after a small number of failed password attempts may result in unacceptable side effects, such

* Version: 15 July 2004. A preliminary version of this paper appears as an extended abstract in the post-proceedings of Financial Cryptography'04 (Springer LNCS, to appear).

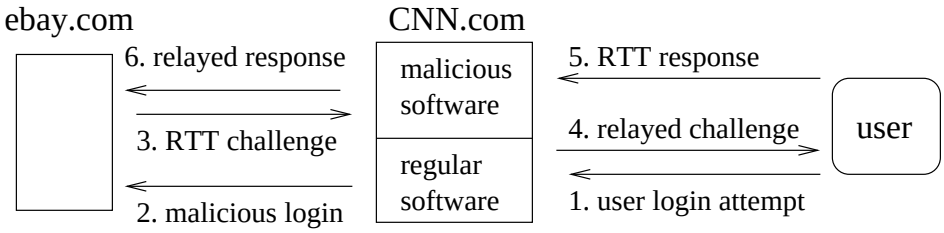

Fig. 1. ATT Relay Attack

as increased customer service costs for additional telephone support related to locked accounts, and new denial of service vectors via intentional lock-out of other users [25]. Another standard approach is to use successively longer delays as the number of successive invalid password attempts on a single account increases. This may lead to similarly unacceptable side effects.

In this paper, we begin by noting that many ATT-based protocols, including that of Pinkas and Sander, are vulnerable to an *ATT relay attack*: ATT challenges may be relayed to possibly unsuspecting parties, who generate responses which are then relayed back to the challenger. We explore this threat and mechanisms to address it, and propose additional (orthogonal) enhancements to the Pinkas-Sander protocol.

The sequel is organized as follows. §2 presents the ATT relay attack. §3 discusses background context and assumptions, including a reference version of the basic ATT-based login protocol. §4 presents a new variation, with enhancements aimed towards usability, security against online dictionary attacks, and parameter flexibility; security analysis is given in §5. §6 pursues per-account customization of the new variation, based on historical user and system statistical profiles. §7 discusses standard techniques to augment general ATT-based login protocols to prevent, detect or deter attacks including the relay attack. §8 provides background and a summary of related work. §9 contains concluding remarks.

Updates. Aside from editorial and minor technical revisions, the primary differences between this paper and an earlier extended abstract are §5 (detailed security analysis), and §6 (tuning protocol parameters based on user profiles).

2 ATT Relay Attack

A relay attack (see §8) may be carried out on online protocols involving an ATT by relaying the ATT challenge to an auxiliary location or "workforce" which generates responses, which are relayed back to the challenger. The original ATT target thus avoids the ATT work.

One attack variant might proceed as follows (see Fig. 1). Assume there are two web sites.[3] The first, say ebay.com, is assumed to be the target of regular online dictionary attacks, and consequently requires correct responses to ATT challenges before allowing

[3] The authors have no affiliation with ebay.com or CNN.com, and no reason to believe either site is insecure. These sites are used as examples simply due to their popularity.

access. The second, say CNN.com, is a popular high volume web site, which for our purposes is assumed to be vulnerable to compromise. The attack begins with an adversary hacking into the CNN.com site and installing attack software.

Upon a user-initiated HTTP connection to CNN.com, the attack software receives the request and initiates a fraudulent login attempt to ebay.com. The attack software, presented with an ATT challenge from ebay.com, redirects it to the CNN.com user connection, instructing that user to answer the ATT to get access to CNN.com. (Many users will follow such instructions: most users are non-technical, unsuspecting, and do as requested.) The CNN.com user responds to the ATT challenge. The attack software relays the response to ebay.com, completing the response to the challenge to the fraudulent login attempt. In conjunction with replying to eBay's ATT challenge, after a sufficient number of passwords guesses (e.g. dictionary attack), an eBay account password can be cracked. The procedure is repeated on other accounts, and the attack program summarizes the online dictionary attack results for the adversary.

The attack is easy to perform if the adversary can control *any* high volume web site – e.g. a popular legitimate site the attacker compromises (as above), or an owned malicious site to which traffic has been drawn, e.g. by illegally hosting popular copyrighted content, a fraudulent lottery, or free software. A related attack involves attack software which relays ATTs to groups of human workers ("sweatshops"), exploiting an inexpensive labor pool willingly acting as a mercenary ATT-answering workforce. An unconfirmed real-world variant reported [22] to involve an "adult web site" requiring users to solve ATTs before being served the content; presumably those running the site relayed the answers to gain access to legitimate sites which posed the original ATT in the hope of preventing automated attacks. Our discussion of mechanisms to counteract such threats continues in §7.

3 Background, Constraints, Assumptions, and Objectives

For reference, Fig. 2 provides a simplified description of the original ATT-based login protocol (for full details, see [20]). The system parameter p is a probability which determines the fraction of time that an ATT is asked, in the case that an invalid userid-password pair is entered. In the case of a successful login, the protocol stores a cookie on the machine from which the login occurred; the cookie contains the userid (plus optionally an expiration date), and is constructed in such a way (e.g. using standard techniques involving symmetric-key encryption or a MAC) that the server can verify its authenticity.

For context, we next state a few assumptions and observations relevant to both the original and new protocols. We begin with a basic constraint.

Constraint 1: Account Lock-out Not Tolerable. We are interested in protocols for systems where locking-out of user accounts after some number of failed login attempts is not a viable option. (Otherwise, online login attacks are easily addressed – see §1.)

```
1  fix a value for system parameter p, 0 < p ≤ 1 (e.g. p = 0.10)
2  user enters userid/password
3  if (user PC has cookie) then server retrieves it
4  if (entered userid/password pair correct) then
5     if (cookie present & validates & unexpired & matches userid) then
6        login passes
7     else % i.e. cookie failure
8        ask an ATT; login passes if answer correct (otherwise fails)
9     endif
10 else % i.e. incorrect userid/password pair
11    set AskAnATT to TRUE with prob. p (otherwise FALSE) †
12    if (AskAnATT) then
13       ask an ATT; wait for answer; then say login fails
14    else
15       immediately say login fails
16    endif
17 endif
```

† This setting is a deterministic function of the userid/password pair [20]

Fig. 2. Original ATT-based Login Protocol (simplified description)

Trust Model Assumptions: Trusted Host and Ephemeral Memory. We assume that client computers, and any resident software at the time of use, are non-malicious (e.g. free of keyboard sniffers and malicious software). This is standard for (one-factor) password-based authentication protocols – otherwise, the password is trivially available to an attacker. For similar reasons, we assume client software leaves no residual data on user machines after a login protocol ends (e.g. memory is cleared as each user logs out). In practice it is difficult to guarantee these assumptions are met (e.g. for borrowed machines in an Internet cafe); but without them, the security of almost all password protocols seems questionable.

Observation 1: Limited Persistence by Legitimate Users. A typical legitimate user will give up after some maximum (e.g. $C = 10$) of failed logins over a fixed time period, after which many will check with a system administrator, colleague or other source for help, or simply stop trying to log in. Large numbers of successive failed logins, if by a legitimate user, may signal a forgotten password or a system availability issue (here login failures are likely not formally recorded by the system); or may occur due to an attacker, as either a side effect of attempting to crack passwords, or intentionally for denial-of-service in systems susceptible to such tactics.

Observation 2: Users Will Seek Convenience. If a login protocol is necessary to access an online service, and users can find a similar alternate service with a more convenient login (though possibly less secure), then many users will switch to the alternate service. User choices are rarely driven by security; usability is usually a far greater factor, and poor usability typically leads to loss of business.

These observations lead us to our usability goal; we state it informally.

Usability Goal – Minimal Inconvenience to Users. Relative to standard userid-password schemes, we wish to minimize additional inconvenience experienced by a user.

As usual, the usability goal must be met in a tradeoff with security, and we have a two-part security goal. One part is protecting specific accounts (e.g. certain users may be more concerned about, or require more, protection; or a service provider may worry more about specific accounts – say those with high sales ratings, or high account values). The second is protecting all accounts in aggregate (e.g. a web service provider might not want *any* user accounts misapropriated to host stolen software; a content service provider might want to protect access to content available to authorized subscribers).

Security Goal – Control Access to both Specific Accounts and Non-specific Accounts. Constrain information the adversary learns from trial password guesses before being "stopped" by an ATT challenge in the context of fully-automated attacks directed towards a specific account (single-account attack) and towards any account (multi-account attack).

In practice, for authentication schemes based on user-selected passwords, prevention of unauthorized access cannot be 100% guaranteed for a specific account or all accounts in aggregate, due to the non-zero probability of correctly guessing a password, and the ubiquity of poor passwords. Nonetheless, the quality of a login protocol may be analyzed independent of particular password choices, and this is what we pursue. For a given password, we are interested in how effectively a given protocol allowing online interaction prevents extraction of password-related information. As little information as possible should be leaked.

Requiring mandatory human participation increases the level of sophistication and resources for an attack. If ATTs are effective and ATT relay attacks are countered (e.g. by means such as embedded warnings – see §7.1), then constraining information leaked before being "stopped" by an ATT challenge is an important security characteristic of a password-based login protocol.

4 History-based Login Protocol with ATT's (New Protocol)

Here we modify the original protocol, intending to both improve the user experience and increase security, e.g. to increase the percentage of time that an adversary is challenged with an ATT, without further inconveniencing legitimate users.[4] The modifications do not themselves prevent ATT relay attacks (§2), but are complementary to those in §7 that do, and can thus be combined.

We assume familiarity with the original protocol (§3; Fig. 2). The new protocol is given in Fig. 3. Line changes from the original protocol are: lines 7.1-7.5 replace 8; and 11.1 replaces 12. The new protocol with failed-login thresholds ($b_1 = 0, b_2 = \infty$) behaves the same as the original protocol.

[4] One might try to improve usability by allowing a small number of trial passwords per userid without triggering an ATT. While this reduces security only minorly for a single-account attack (§5.1), the problem is greater with multi-account attacks (§5.2).

```
1   fix values for 0 < q ≤ 1 (e.g. q = 0.05 or 0.10) and integers b1, b2 ≥ 0
2   user enters userid/password
3   if (user PC has cookie) then server retrieves it
4   if (entered userid/password pair correct) then
5     if (cookie present & validates & unexpired & matches userid) then
6       login passes
7     else % i.e. cookie failure
7.1     if OwnerMode(userid) OR (FailedLogins[userid] ≥ b1) then
7.2       ask an ATT; login passes if answer correct (otherwise fails)
7.3     else
7.4       login passes
7.5     endif
9     endif
10  else % i.e. incorrect userid/password pair
11    set AskAnATT to TRUE with prob. q (otherwise FALSE) †
11.1  if (AskAnATT) OR (FailedLogins[userid] ≥ b2) then
13      ask an ATT; wait for answer; then say login fails
14    else
15      immediately say login fails
16    endif
17  endif
  † This setting is a deterministic function of the userid/password pair [20]
```

Fig. 3. New Protocol (History-based Login Protocol with ATT's). OwnerMode(userid) is a boolean function (e.g. implemented by table-lookup into a bit-array) returning TRUE if the associated account is in owner mode; its update is not shown. *FailedLogins[userid]* is set to the userid's number of failed logins in a recent period T, and updated (also not shown). See §4 re: handling cookies and definition of owner mode.

We next discuss some differences between the new and original protocols, including: cookie-handling (related to owner and non-owner mode) – cookies are now stored only on "trustworthy" machines; per-user tracking of failed logins; and setting failed-login thresholds. The idea of dynamically changing failed-login thresholds has been previously mentioned [20, §4.4-4.6]; we detail a concrete proposal and comparison.

Handling cookies. The original protocol stores a cookie on any device after successful authentication; the new protocol does not. Optional user input controls cookie storage similar to web servers using a login page checkbox asking if users want to "remember passwords", e.g. "Is this a trustworthy device you use regularly? YES/NO". This part of the page appears if no cookie is received by the server. Upon a YES response, a cookie is pushed to the user device only after the user successfully authenticates (requiring a successful ATT response, if challenged). This cookie approach reduces exposure to cookie theft vs. the original protocol, with negligible usability downside because the question appears on the same screen as the login prompt (default answer NO).

The original protocol requires that cookies be tracked by the server and expire after a limit of failed login attempts with the particular cookie (e.g. 100 [20, §4.5]). We follow a similar approach. Each time a login fails (e.g. lines 7.2, 13, and 15), we increment the failed

login count associated with the cookie if a valid cookie was received. If the cookie exceeds a failed login threshold, we invalidate it. Line 5 includes a check that the cookie hasn't been invalidated. The *cookie failure threshold* is the number of failed logins allowed before a cookie is invalidated. We recommend setting this to $\min(b_1, b_2)$.

Definition of owner, non-owner. A user is more likely to login from "non-owned" devices when traveling (e.g. borrowing an Internet access device in a library, guest office, conference room, or Internet cafe). Also, a user submitting a login request without valid cookie is likely using a non-owned device. As a consequence of how cookies are handled, we can assume (with small error) that a user is on a non-owned device if their most recent successful login is cookieless. We initially define a user account to be in "*owner*" mode, and expect an account to be in owner mode most of the time if most of the time users use their regular device (e.g. one of the devices they own). An account transitions to "*non-owner*" mode when a login is successfully authenticated without the server receiving a valid cookie (Fig. 3, line 7.4), and returns to owner mode after a specified time-out period W (e.g. 24 hours) or a successful login with cookie present. The timeout period is restarted, and the account remains in non-owner mode, if another cookieless successful login occurs. The time-out period reduces the number of accounts in non-owner mode, which lowers the security risk; accounts in non-owner mode are more susceptible to multi-account dictionary attacks (see §5.2).

Tracking failed logins. We define *FailedLogins[userid]* to be the number of failed login attempts for a specific userid within a recent period T (e.g. 30 days). Here *failed login attempts* includes: non-responses to ATT challenges, incorrect responses, failed userid-password pairs, and outstanding authentication attempts (e.g. the adversary may simultaneously issue multiple login attempts; one strategy might be to issue a very large number, and respond to only a subset of resulting ATT challenges, perhaps being able to exploit some "weak sub-class" of ATTs for which computer-generated responses are feasible).

Setting the failed-login thresholds (bounds b_1, b_2). Low values for b_1, b_2 maximize security at the expense of usability (e.g. for users who frequently enter incorrect passwords). A reasonable bound may be $b_1, b_2 \leq 10$ (perhaps larger for large T). In the simplest variation, the protocol bounds b_1, b_2 are fixed system variables; in a more elaborate design (cf. §6), they are dynamic and/or set on a per-user basis (varying for a particular userid, based on a history or profile and possibly subject to system wide constraints e.g. maximum bound on b_2). For example, certain users who regularly enter a password incorrectly might be given a higher failed-login threshold (to increase usability) compared to users who almost always enter correct passwords. If it is expected or known from a historical profile that a user will log in L times over a period T, and that say 5% of legitimate login attempts fail, then b_2 might be set somewhat larger than $(0.05) * L$ (e.g. $T = 30$ days, $L = 100$, $b_2 = 5$). Over time, per-user rates of legitimate failed logins (e.g. mistyped or forgotten/mixed up passwords, perhaps more frequent on unfamiliar machines) can be used to establish reasonable thresholds. To simplify presentation, updating of per-user table entries *FailedLogins[userid]* in Fig. 3 is not shown. Note that while per-user values require server-side storage when these values cannot

Question	Original Protocol	New Protocol	
		Account Mode	
		Owner	Non-owner
Q1	$(1-p)N$	$(1-q)b_2$	$\max(b_1,(1-q)b_2)$
Q2	$\frac{1}{2}pN$	$\frac{1}{2}(N-(1-q)b_2) \approx N/2$	$\frac{1}{2}(N-\max(b_1,(1-q)b_2))$
Q3a ($c=0$)	0	0	b_1/N
Q3b ($c=1$)	$1/pN$	$(1-(1-q)^{b_2+1})/qN$	$(b_1+1)/N$
Q3c ($c \geq 2$)	c/pN	$\leq \min(\frac{c}{q}, b_2+c)/N$	$(b_1+c)/N$

Table 1. Summary of comparitive security analysis (single-account attack).

be user-stored via cookies, a small amount of per-user server-side storage is already required in both the original and new protocol to ameliorate cookie theft (see above). (Optionally, setting the ATT challenge probability q on a per-user basis also allows flexibility for tuning usability and security on a per-account basis.)

5 Comparitive Analysis – Security and Usability

In this section we provide a comparitive analysis of the new and original protocols. We begin with a security analysis against single-account attacks, then consider multi-account and other attacks, and usability. We generally follow the original assumptions [20], including that passwords are from a fixed set (dictionary) of cardinality N, and that for analysis purposes these passwords are equi-probable – nonetheless acknowledging that the latter, in particular, is false in practice. Probabilities p and q are as defined in the protocols. Below, we first assume no cookie theft (an attacker knows a userid but has no corresponding cookie, so that a correct password guess puts the attacker into the cookie failure block at line 7 in the new protocol), but in §5.3 consider the implications of theft. We make an additional simplifying assumption: a particular account remains in either owner or non-owner mode.

5.1 Security Analysis (single-account attacks)

To drive our comparitive security analysis, we ask for the original and new protocols, for an attack on a single account, what is the:

Q1: expected number of passwords eliminatable from the space, answering no ATT's?
Q2: expected number of ATT's an attacker must answer to correctly guess a password?
Q3: probability of a confirmed correct guess for attacker willing to answer c ATT's?

Table 1 summarizes the answers based on the best attack strategies known to the authors (see next paragraph), and the assumptions noted above. Also, to the benefit of the attacker, we assume that failed login counts b_1 and b_2 are 0 at the start of an attack. In the table, q is used for p in the new protocol to emphasize $q \neq p$ is allowed.

For Q1, in a first pass, an attacker of the original protocol may simply try all passwords in an attack dictionary, and quit on each guess triggering an ATT (in total, a p-fraction of the dictionary); all others, i.e. $(1-p)N$ passwords, result in a protocol response confirming password incorrectness without the cost of answering an ATT, and thus can be eliminated at a cost of zero ATTs. In the new protocol, the number of "free" (i.e. without ATT) guesses in owner mode is certainly limited by the failed-login threshold b_2, after which an ATT is required for each guess; but for a q-fraction of these b_2 guesses, one expects an ATT triggers at line 13, which to the attacker is indistinguishable from a line 7.2 ATT – thus these passwords cannot be safely discarded as incorrect. Thus the expected number of eliminatable passwords here is actually $(1-q)b_2$, as noted in the table. Similar reasoning yields the table entries for Q1 and Q2 in non-owner mode.

For Q2, an attacker would on average be expected to guess the correct password after going through half the remaining (non-eliminated) passwords, and each of these guesses has a cost of one ATT. Thus e.g. in the new protocol for an owner mode account, after eliminating $(1-q)b_2$ passwords, an attacker is expected to try half the remaining $N-(1-q)b_2$ passwords, at a cost of $(N-(1-q)b_2)/2$ ATTs, before hitting the correct password. Similar reasoning gives the Q2 table entry for the original protocol.

For Q2 and Q3 against both the original protocol and owner-mode accounts in the new protocol, for an attacker who measures cost in terms of the number of ATTs that must be answered, there appears to be no penalty in simply answering the first c ATT's asked (without eliminating bad passwords in a preliminary "zero ATT cost" pass). The reason is as follows. In the original protocol, ATTs are asked on only a p-fraction of the password space, which includes the correct password; and upon being asked an ATT, there is a fixed probability that the password candidate is correct. For the new protocol, the same holds, but in addition, each non-answered ATT increases the failed-login counter, which shortens the number of remaining guesses available (if any) before triggering an ATT on every single guess. In essence, an attacker trying a candidate password W gains no benefit from failing to pursue W upon getting an ATT (unless he has no intention of answering *any* ATTs, which is a valid multi-account attack strategy against non-owner mode accounts – see §5.2); rather, the ATT should be answered since W may indeed be the one correct password, and subsequent trials of the same password W will also trigger an ATT.

The answer to Q3 for the original protocol is c/pN. This follows from being able to narrow the password space down to pN candidates, and c ATTs allowing the fraction c/pN of this space to be covered. For $c=0$ in owner mode of the new protocol, the probability is 0 of correctly guessing a password without answering an ATT, since an ATT is always asked if the correct password is tried without a valid cookie. For $c=0$ in non-owner mode, an attacker has b_1 "free" guesses before the failed-login threshold kicks in at line 7.1, and any ATT asked during the first b_1 trials necessarily results from line 13, confirming an incorrect password; furthermore any password guess after trial b_1 cannot be confirmed as correct, since line 7.1 would trigger an ATT which the attacker is not willing to answer. This gives a

Q3a probability of b_1/N with 0 ATTs, and similarly for c ATTs, a probability of $b_1 + c/N$. The remaining entries in the table, namely Q3b ($c = 1$) and Q3c ($c \geq 2$) for owner mode of the new protocol, require further explanation, as given by the following Lemmas.

Lemma 1. Let R be the probability of a confirmed correct guess for an attacker willing to answer $c = 1$ ATT in owner mode of the new protocol. Then $R = (1 - (1 - q)^{b_2+1})/qN$.

Proof: Arguing as above with an attacker simply answering the first ATT asked, the number of trial password guesses is limited to $b_2 + 1$, since testing bound b_2 at line 11.1 guarantees a second ATT if the game ever proceeds beyond $b_2 + 1$ trial guesses, with each such trial thus having zero probability of success. So R is the sum of the probabilities of a confirmed correct guess over the first $b_2 + 1$ trials. Let c_i be the probability of a correct guess upon reaching trial i, and let e_i be the probability the event trial i occurs. Then $R = \sum_{i=1}^{b_2+1} c_i \cdot e_i$. Now for all i, it follows from simple probability tree (or other) arguments that $c_i = 1/N$. Regarding e_i, for trial i to occur, line 13 must have been avoided on all previous iterations (otherwise the single ATT would have been consumed); this has probability $(1 - q)^{i-1}$. Thus $R = (1/N) \sum_{i=1}^{b_2+1} (1 - q)^{i-1} = (1 - (1 - q)^{b_2+1})/qN$, as claimed.

Note 1. From Lemma 1, if $b_2 = 0$ then $R = 1/N$; this yields a smaller (better) probability than the corresponding $1/pN$ in the original protocol. Moreover, the bound b_2 check in line 11.1 ensures at most $b_2 + 1$ trials can pass before consuming the $c = 1$ available ATT, so for any value b_2, $R \leq (b_2 + 1)/N$ (at most $b_2 + 1$ of N passwords can be guessed, using in total one ATT). Finally, as $b_2 \to \infty$, $(1 - q)^{b_2+1} \to 0$ for $q > 0$, and thus $R \to 1/qN$, as in the original protocol; this is as expected since $b_2 \to \infty$ means the bound b_2 becomes unused. Combining these we have $R \leq \min(1/qN, (b_2 + 1)/N)$, with the first term $1/qN$ taking precedence when $q \geq 1/(b_2 + 1)$, in which case larger probabilities q lower R.

Lemma 2. Let R be the probability of a confirmed correct guess for an attacker willing to answer $c \geq 2$ ATTs in owner mode of the new protocol, using an attack strategy of answering the first c ATTs asked. Then $R \leq \min(\frac{c}{q}, b_2 + c)/N$.

Proof: Similar to reasoning in the proof of Lemma 1, checking (only) the threshold b_2 in line 11.1 implies that at most $b_2 + c$ passwords can be guessed before c ATTs are consumed. (In fact one would expect fewer trials, due to the probability q in line 11 also contributing to the consumption.) This yields the $(b_2 + c)/N$ term. The c/qN term follows from the fact that the probability q (alone) at line 11 would cause line 13 to be expected to occur every $1/q$ trials, and thus in the absence of a b_2 bound one would expect line 13 to consume c ATTs after c/q trials, allowing the attacker to guess c/q of the N password candidates, giving an expected success probability of c/qN. The interaction between these two possible outcomes, each of which consumes ATTs independently, gives the claimed upper bound.

Note 2. From Lemma 2, the second term $(b_2+c)/N$ takes precedence provided $q \leq c/(b_2+c)$, which we would expect for many practical parameter selections (when $c \neq 0$). For a bound tighter than that given by Lemma 2, and better understanding of the protocol, we consider a game between the algorithm and the attacker, with the attacker willing to "spend" c ATTs, answering each ATT when asked. The question is to find g, the expected number of trial password guesses by the attacker, up to and including the one triggering the last (c^{th}) ATT; it then follows that the probability of a successful password guess is g/N. We find it helpful to consider two cases. Case 1: the c^{th} ATT is consumed before the point at which the bound b_2 triggers an ATT at every single iteration of line 11.1 – this will happen for $c/q \leq b_2$, probably an uncommon parameter choice (e.g. $q = 0.1$, $c = 2$, $b_2 = 25$). Case 2: the c^{th} ATT is consumed after trial b_2, but within the first $b_2 + c$ trials.[5]

Determining g, and hence the answer to Q3c (new protocol, owner mode), thus motivates the question: what is the number x of trials (password guesses) expected before consuming the c^{th} ATT? Under our assumptions discussed earlier, an ATT results from line 11 of the new protocol with independent probabilility q on each trial, assuming an attacker randomly selects (different) password candidates. This question is recognizable as a standard negative binomial distribution question, with the random variable X being the number of "failures" (trials where no ATT is asked), the experiment continued until a fixed number c "successes" occur (an ATT is asked), the probability of success and failure respectively being q and $1 - q$, and our interest being in the expected value of $g = x + c$, for the probability function $f(x) = C(x + c - 1, x) \cdot q^c(1 - q)^x$ for $x = 0, 1, \ldots$. ¿From this a precise expression for R in Lemma 2 can be given, but a general closed form is not evident; thus we find the bound given by Lemma 2, and the discussion above, to be preferable.

Note 3. Rows Q1 and Q2 of Table 1 indicate that the number of passwords that an attacker can eliminate "for free" (without any ATT's) is substantially greater in the original protocol.[6] A second observation favoring the new protocol follows from rows Q3b and Q3c: the probability of a successful attacker guess in the new protocol (on the order of $1/N$) is generally significantly smaller than in the original (on the order of $1/pN$), except that when b_2 is relatively large the new protocol effectively behaves as in the original, with probability c/qN matching the table entry c/pN for the original protocol; when b_2 is small, $b_2 + c$ is less than c/q, so the probability in the new protocol is better, i.e. less than in the original.

Note 4. Row Q3a indicates that for an attacker unwilling to answer any ATTs, both protocols have the same security, except that we relax security in the new protocol (i.e. to b_1/N)

[5] Clearly, after trial number $b_2 + c$, a correct password guess cannot be confirmed, as all c ATTs have been consumed. As per Note 2, b_2 plays a role in limiting the number of password guesses iff $c/q \geq b_2 + c$.

[6] For the new protocol, these figures are per time period T. However for a sophisticated multi-period attack, the new protocol remains better (fewer passwords are eliminatable), assuming $p = q$, unless at least N/b_2 time periods are used (e.g. about 1600 years for $T = 1$ month, $N = 100\,000$ and $b_2 = 5$).

for those accounts in non-owner mode (presumably a relatively small number), to improve usability as indicated in Table 2 (bottom row).

Note 5. For both $c = 1$ and $c \geq 2$, Table 1 gives a probability (i.e. an expectation over a large number of runs), for which the new protocol has a guaranteed upper bound (i.e. worst case) of $(b_2 + c)/N$. For reasonable values b_2, this is substantially better (lower) than the corresponding expected value in the original protocol.

5.2 Security Analysis (multi-account attacks)

In multi-account (or system-wide) attacks, an attacker seeks to break into any one of many accounts, not necessarily a specific account. They are generally of greater concern to service-providers than individual users, and usually more difficult to protect against (than single-account attacks) since they tend to find "weakest links" across a system or user space.

In the analysis below we use assumptions from the beginning of §5, and here in addition assume that an attacker knows m valid userids of the L total user accounts; setting $m = L$ gives the attacker the greatest advantage, and in this sense is the most general (worst) case – albeit over-estimating insecurity in some environments.

Case $c = 0$ (zero ATT's answered). For the new protocol, consider an attack strategy where the attacker (assumed to have no valid cookie) acquires a list of m valid userids for accounts in non-owner mode (or exhaustively tries userid-password combinations hoping to meet this condition – see Note 6) and makes password guesses. Either a guess is correct ("lucky guess" resulting in line 7.4 in Fig. 3), it is incorrect with an immediate failure notice given (line 15), or an ATT challenge results (lines 7.2 or 13). The attacker quits all login sessions returning an ATT, moving onto a new password (on the same userid at most b_1 times, or another userid). The goal is to reach line 7.4; note this requires an account in non-owner mode, and below the failed login bound b_1. For accounts in owner mode, the new protocol performs similarly (from a security perspective) against multi-account attacks as the original protocol.

For non-owner mode accounts, the new protocol permits an attacker b_1 guesses (per account) before challenging with an ATT on any subsequent *correct* password guess. In contrast, in the original protocol an ATT challenge occurs every single time a password guess results in line 8, and thus the probability of successful attack (for $c = 0$) is zero. Thus in non-owner mode, security has been decreased, in a trade-off for improved usability (see §5.4). This security loss can be made arbitrarily small by reducing b_1, e.g. at the extreme setting $b_1 = 0$; this has the effect of artificially putting the account into owner mode, yielding behaviour matching that of the original protocol at line 8 in Fig. 2.

Note that accounts found, or forced to adhere to a strong password selection policy (e.g. through password checking programs that check the quality of passwords and/or password

generation programs [9, 10]) could be assigned a setting $b_1 \geq 1$ or somewhat higher, with $b_1 = 0$ (or very low) for other accounts. Such tuning of b_1 on a per-user basis (see related discussion in §6) effectively results in users who chooose "weak" passwords forfeiting usability benefits in non-owner mode. Alternatively, users of accounts which are often in non-owner mode could be selectively encouraged to use stronger passwords.

Case $c \geq 1$ (one or more ATTs answered). For the new protocol with an attacker willing to answer one or more ATTs, the best strategy would again seem to involving making trial guesses on a particular non-owner mode account until reaching the failed-login bound b_1 for that account, and then moving on to another non-owner mode account. This attack does not succeed against accounts in owner mode, since for such accounts, an ATT is demanded (guaranteed) upon reaching line 7.1.

As discussed earlier (Table 1, row Q3c), the new protocol has significantly better security than the original against *single-account* attackers willing to answer $c \geq 1$ ATTs. This security advantage carries over for multi-account attacks (including in situations where the adversary arranges that ATTs be answered e.g. by relaying them to a "sweatshop"), but only for accounts in owner mode. For non-owner mode accounts, it is thus important to take special pre-cautions as suggested under Case $c = 0$. This leads to Note 6.

Note 6. It is of significant advantage to an attacker of the new protocol (Fig. 3) to find ways to distinguish owner from non-owner mode accounts; and correspondingly, for the system to prevent this information from becoming easily available. An attacker unable to distinguish the two will face far more ATTs, resulting in higher system-wide security; and relatedly, the system ratio of owner vs. non-owner accounts affects security from the viewpoint of system resistance to multi-account attacks.

5.3 Other Attacks

Here we briefly consider several other types of attack.

Parallel login attacks. An attacker may try to launch a parallel-login attack, simultaneously attempting a login to one userid a large number of times (e.g. a thousand) on different servers. (See related definition of failed login attempts in §4.) This attack attempts to take advantage of architectures which involve large number of servers to load-balance activities in the case of large user spaces, and the difficulty of centrally updating failed login counts (or other authentication state information) across different servers. It can be countered by routing authentication requests by userid, to a particular server or server farm pre-assigned to that userid. With such a system design, each such server is able to stay current on updates without excessive login delays.

Cookie theft. The above analysis assumes no cookie theft occurs. Here we make a few observations in case it does.

New Protocol. If a cookie is stolen, then within the cookie's validity period and while under the recommended cookie failure threshold (see §4 and paragraph below), the attacker can try $\min(b_1, b_2)$ password guesses on the associated userid, without being asked an ATT that cannot be prudently abandoned, using a single-account attack as follows. The attacker guesses passwords, quits all guesses that return an ATT, and hopes to reach line 6 (Fig. 3) with a lucky guess. The probability of success is $\min(b_1, b_2)/N$. (This attack may take place in conjunction with one that reduces the password space without answering an ATT, or one where the adversary is willing to answer $c \geq 1$ ATTs.)

Original Protocol. Similarly, the attacker gets free guesses up to the cookie failure threshold. A correct password guess on any of these trials allows a successful login without having to answer an ATT.

Note 7. (a) Stolen cookies are less likely with the new protocol, since they reside in fewer places – e.g. cookies from the original protocol would show up in airport Internet rooms. While any device can become easily compromised by malicious software exploiting any of thousands of software flaws, under the assumption that the vulnerability to cookie theft is proportional to the number of cookies on devices, the original protocol is more vulnerable than the new to cookie theft. (b) A combined cookie and non-cookie attack against a single account has lower probability of success in the new protocol, due to the failed login thresholds; moreover, in the original protocol, the attacker can reduce the password space to a p-fraction even before using the stolen cookie (cf. discussion on Q1 and Q2 in §5.1).

5.4 Discussion of Usability

For comparing usability between the original and new protocols, Table 2 notes the proportion of time a legitimate user is queried with an ATT on entering a correct or incorrect password, with and without a valid cookie. A case of particular focus for the new protocol is the legitimate "travelling user", who generally operates with an account in non-owner mode and without a valid cookie. The new protocol is significantly more user-friendly to such users. We also believe that such users are typically more likely to enter incorrect passwords (see Note 8), and therefore increasing usability in this case is significant as one would expect that "incorrect password" cases occur far less often in owner mode.

Also related to usability – the value of the parameter q may be reduced in the new protocol without loss of security, due to the use of the failed login bound b_2 and depending on its value relative to q (see Table 1). This further increases usability in the incorrect password case, independent of the discussion in the paragraph above.

Note 8. Regarding Table 2, after the failed login bound is crossed in the new protocol, in several cases – e.g. on incorrect passwords, and correct passwords without valid cookies

	Original Protocol	New Protocol	
		Account Mode	
		Owner	Non-owner
Incorrect password	p	q†	q†
Correct password - valid cookie	0	0	0
Correct password - no valid cookie	1.0	1.0	0†

Table 2. Fraction of the time a legitimate user must answer an ATT (1.0 = 100%). †See Note 8.

– ATT's occur more frequently (i.e. 100% of the time after the bound is crossed within period T). However for accounts in owner mode we expect a large number of users select a "Remember password" option (standard in many applications) which stores passwords locally on their regular machines. No failed passwords are expected from such users; but note their failed login thresholds may still be crossed due to attacker activities.

6 Tuning Protocol Parameters with User Profiles

We enhance the basic version of the new protocol by adjusting protocol parameters based on historical user profiles (cf. §4), i.e. statistics derived from prior login attempts. This allows per-account tailoring of security and usability, improving overall system security against multi-account attacks, and reasonable usability adjustments for different types of users.

6.1 Single and Multi-Account Historical Statistics

In the definitions of characteristics comprising the historical user profiles below, we use the following terms. *Login session* means a sequence of one or more login attempts to a particular account within some fixed time window, and which can reasonably be attributed to the same source – e.g. originating from the same network address, or supplying the same cookie, within a 5-minute time period. *Successful login session* means a session resulting in account access, whereas a *failed login session* does not. Suggested account characteristics comprising an account profile include the following.

1. **Password Failure Rate.** An account's "password failure rate" is the *average number of failed login attempts per successful login session.* Excluding failed login sessions here limits an attacker's ability to manipulate password failure profiles by submitting invalid passwords to user accounts. As one variation, this statistic is tracked separately for owner and non-owner mode, to allow tuning if users employ mechanisms for reliable password entry on owned machines (e.g. browser auto-fill features). As a second, this statistic is tracked separately for valid-cookie and cookieless successful sessions, i.e. separating those sessions which involve receipt of a valid cookie.
2. **Borrowing Rate.** An account's "borrowing rate" is the *ratio of successful login sessions without submitting a valid cookie to those with a valid cookie.*

3. **Group Failed Login Count.** For a set of accounts, the "group failed login count" is the *total number of failed logins for the user accounts in question, over some time window, associated with failed login sessions involving no valid cookie.*
   The set of accounts may e.g. be the entire account space, a statistical sampling, or some number of subsets of accounts viewed as attractive targets (e.g. financial accounts of high monetary value, or accounts recently active in an online auction).

The above statistics are computed over (one or more) reasonably chosen recent time windows, long enough to model the predominant failure modes. For example, to account for password failures due to infrequent logins (with users simply forgetting passwords), the window for password failure rate could be set on the order of months or more.

6.2 Protocol Enhancements Using Historical Statistics

We now suggest a number of custom enhancements to usability and security, by dynamically adjusting protocol parameters based on historical profiles.

1. **(non-owner mode accounts:) in absence of a valid cookie, set b_1 proportional to the password failure rate for invalid cookies.**
   Justification: For accounts in non-owner mode, a valid cookie not being received is consistent with the user continuing to travel and log in from a borrowed machine. This customization improves usability for the case where the user has a history of password failures when traveling. The refinement on cookie validity addresses variability in password failure rates based on whether the user is on an untrusted or trusted machine (the latter being identified by requests accompanied by a valid cookie) – and a significant difference is expected, in part due to the use of browser features that store and suggest the username and password associated with a particular web page; these tend to reduce password failures rates from trusted machines, and amplify failure rates on untrusted machines (as the user must recall an even less frequently used password).
2. **(owner mode accounts:) set b_1 proportional to the borrowing rate.**
   Justification: For accounts in owner mode, the value of b_1 is relevant when the user begins to travel, and the account is to transition from owner to non-owner mode. For accounts with a zero (or very low) borrowing rate, setting b_1 to zero provides maximum security, at the expense of requiring the user to answer an ATT in the (unlikely) case of using a borrowed device. Recall $b_1 = 0$ forces an ATT (Fig. 3, line 7.2) challenge upon valid userid-password entry without a valid cookie – the latter being characteristic of both logging in from a borrowed device, and an attack whereby the attacker does not have access to cookies stored on a trusted machine. Setting b_1 proportional to the borrowing rate (with a suitable upper bound) is consistent with the reasoning that leniency (i.e. user-friendliness in the form of fewer ATT challenges) should depend on the chances that a legitimate user is making the login request.

3. **(owner mode accounts:) on receipt of a valid cookie, set b_2 proportional to the password failure rate.**
   Justification: If an account is historically prone to password failures, and a valid cookie is received, then it may be likely that the login attempt came from a valid user. For users prone to login errors, usability is increased by selectively increasing b_2.
4. **increase q (for group members) if the group failed login count rises substantially, and in this case also decrease b_1 and b_2.**
   Justification: Such an increase may suggest a multi-account attack, in which case lowering b_1 and b_2, and raising q, will make the attacker's task more dificult (at the cost of increased inconvenience to associated users).

7 Additional Techniques Augmenting ATT-based Authentication

Here we propose a number of techniques to augment the original protocol (Fig. 2), without changing its basic functionality. This includes addressing ATT relay attacks (§2). These techniques are intended primarily to improve security, and are independent of (complementary to) the changes proposed in §4. We present them briefly without additional analysis.

7.1 ATT with Embedded Warning

Here we propose a simple method to prevent ATT relay attacks. A drawback of the proposal is that it requires some thought on behalf of users (which is, in some cases, unfortunately unrealistic). However, we believe the general idea may be adapted to significant advantage.

The general idea is to rely upon self-awareness of legitimate users to prevent unwitting participation in an ATT relay attack. One approach is to make ATT challenges user-directed by incorporating a user's specific userid *within the ATT itself*. Preferably, removing this information is of comparable difficulty as answering the ATT itself.

For example, as part of answering a text ATT, a portion of the text is a userid field,[7] which the user is warned to compare to their own userid, thereby confirming that the ATT is targeted specifically at them (within the embedded warning the user is instructed to not answer the ATT if the match fails). As an additional optional feature, the ATT might also contain an embedded short "help URL", for a site giving further instructions on the use of this type of ATT.

This idea is analogous to the now generally accepted, and recommended, practice in authentication protocols of putting party identifiers within the protected (i.e. signed or MAC'd) region of protocol messages. It is also analogous to the typical automated check, when using secure browser cookies, that cookies match a particular userid or IP address; and to the matching userid check in the original protocol (line 5, Fig. 2).

[7] A variant instead includes the name of the site being visited, with similar explanation. (An anonymous referee suggested this.) The choice between web site name and userid could be made dynamically, e.g. selecting the shorter of the two.

7.2 Notification Regarding Failed Logins

Here we propose a simple method to detect automated dictionary attacks and trigger counter-active measures.[8] Once a small threshold (e.g. 3-10) login failures occurs for any single account, an automated, out-of-band communication (e.g. email) is sent to an address-on-record of the associated legitimate user. If the failed logins resulted from the user's own actions, the user will be aware of the failures and can safely ignore the message; otherwise, it signals malicious activity, and may lead the user to take such actions as to request[9] changes to server-side user-specific login protocol parameters (see §4), or to change their own password to a more secure password using the normal change password method.

As an alternative, albeit less desirable,[10] after some larger number of failed logins (e.g. 25), the system might automatically reset the user's password to a computer-generated secure password emailed to the user. This would prevent a user's typically weak self-chosen password from being cracked through standard dictionary attacks. (Depending on the security policy in use, the user might be allowed to change the password back to a weak one if they wish, but at this point they may also be motivated to follow recommended password rules.)

This proposal is less effective against multi-target attacks, and *slow-channel dictionary attacks* wherein an automated program tries passwords on a certain account after there is likely to have already been a successful login attempt (e.g. waiting for a random but minimal delay, such as one-day intervals). In some systems, an attacker can confirm if a user has logged in recently (e.g. an eBay user), and mount only a limited number of trial password guesses some fixed period after each such successful login. This proposal may nonetheless be helpful, and other parameters may limit the success of slow-channel attacks. A small amount of per-user server-side state is needed, but the original protocol has a similar requirement to address cookie-theft [20, §4.5]. A remaining drawback of this proposal is degraded usability (additional user attention is required).

7.3 Consuming Client Resources using Zero-Footprint Software Downloads

We propose that login protocol variants (e.g. see §4) be augmented by known techniques requiring that clients solve "puzzles" consuming client resources, and return answers prior to the server verifying a login. This follows research lines to combat junk mail (e.g. [8, 1]) and denial-of-service attacks [15]. Another augmenting technology is to harden passwords with auxilliary protocols that can interact directly with the server [11].

---

[8] This expands on administrators manually sending out-of-band messages [20, §4.4].

[9] For example, through an authenticated channel such as an email to an un-advertised pre-arranged address, or a hidden URL provided in the email alert to the user.

[10] This may raise customary issues related to system-generated passwords and system-initiated password changes. If used, this alternative must be crafted so as not to generate additional customer service calls, which are not tolerated within our scope.

Since functionality for performing client puzzles is not resident in standard client software (e.g. browsers), this proposal requires allowing Java applets, Javascript, or other zero-footprint downloads. We no longer agree with dismissing special client-side software outright (cf. [20]); rather, we see opportunity for advantageous use. Though perhaps worrisome, most users and organizations now operate under the assumption that Java, and certainly Javascript, are turned on.[11] Nonetheless, since popular web services should work for 100% of potential users, to accommodate those who cannot use zero-footprint software, ATT-based login protocols can be designed as follows. Client puzzles (or the like) are sent to users. For those unable to answer the puzzles for any reason (in some case the server may learn this *a priori*), the protocol branches to a path replacing the puzzle by an (extra) ATT. This ATT will be less convenient to the user (requiring user attention, vs. machine resources), but we expect this to be a relatively small percentage of users, and therefore viable.

Another approach to strengthening login protocols involves "strong authentication protocols" like EKE (see §8), which in general would also require extra client-side software. However, these techniques do not appear to be of use for our problem. EKE-like protocols are designed to preclude off-line (vs. online) dictionary attacks, and typically for systems with passwords of very low entropy, which thus rely on account lock-out (as very low entropy passwords can be cracked in a relatively small number of guesses). In contrast, we are interested in environments where account lock-out is not viable.

8 Background and Related Work

The ATT relay attack of §2 is related to general classes of *middle-person attacks* and *interleaving attacks* involving an active attacker inserting itself between legitimate parties in a communications protocol, and/or using information from one instance of a protocol to attack a simultaneous instance. Such attacks are well-known in cryptographic protocols and have a long history ([6, 7]; [18, pp.530-531]).

For example, challenge-response protocols have long been used to identify military aircraft in *identify-friend-or-foe* (IFF) systems. IFF challenges from enemy challengers have reportedly been forwarded in real-time to the enemy's own planes, eliciting correct responses which were then successfully used as responses to the enemy's original challenges [2, pp.19-20]. Note that responses in such systems are typically automatic; the protocols do not involve entity authentication of the querying party.

Related to this is the well-known grandmaster postal-chess attack: an amateur simultaneously plays two grandmasters by post, playing white pieces in one game and black in the other, using the moves of his opponents against each other, resulting in an overall outcome better than the two losses he would have achieved on his own.

[11] These are in fact the settings that result from the Internet Explorer default ("medium" security), and which we expect remain unchanged by most users.

The term *strong authentication protocols* is often used for protocols designed to preclude attacks which first obtain appropriate data related to one or more protocol runs, and then proceed to crack passwords offline (i.e. without further interaction). This line of research began with the early work of Gong and co-authors [17, 12, 13]; Bellovin and Merritt's EKE protocol [3] then inspired a number of others (e.g. Jablon's SPEKE [14]; Wu's SRP [24]; see also [16]).

Offline exhaustive password-guessing attacks typically proceed by trying potential passwords in order of (perceived) decreasing likelihood. The most probable passwords are often in conventional dictionaries, or modified dictionaries specially tailored to this task. Offline attacks are thus often called *dictionary attacks*, although dictionaries are also used in online attacks (if account lock-out and time-delays are not used; see §2).

Use of system-generated passwords can provide higher security (by better password choices), but suffers severe usability issues. Pass*phrases* have also been proposed (e.g. see [28, 27]). Other approaches include system administrators running password-crack tools on their own systems (*re-active* password checking); enforcement of simple password rules or policies at the time of new password selection; and at such time, checking for its presence in large customized dictionaries built for this purpose (*pro-active* password checking, e.g. see Yan [26] for a recent summary).

9 Concluding Remarks

We expect that a large number of human-in-the-loop and mandatory human participation schemes, unrelated to the ATT-based login protocol discussed here, are also subject to the ATT relay attack of §2.

A major feature of our new protocol (§4) is the additional flexibility and configurability, including failed login thresholds and potentially lower ATT challenge probabilities (e.g. for suitable b_2 lowering q does not decrease security). This allows the protocol to be tailored to match particular environments, classes of users, and applications; while determining the optimal parameters for specific user profiles appears non-trivial, we expect further analytical study will be fruitful. Another new aspect is storing cookies only on trustworthy machines. As mentioned in §4, the new protocol can be parameterized to give the original protocol as a special case. While the configurability does complicate protocol implementation somewhat, we note that a number of the parameters which are optionally dynamic can be managed by automated tools; thus the additional human administrative costs are relatively minor. For example, an automated tool can keep a running ratio of successful logins to failed logins for the entire system, and alter system wide (or account-specific) parameter q, or system-wide (or account-specific) failed login thresholds b_1 and b_2, based on this ratio. A significant improvement of our protocol over prior work concerns protecting against relay attacks by forcing an ATT challenge on all login attempts after the number of failed logins reaches a threshold. Previous work enabled a significant fraction of the password space to

be eliminated with an automated attack. Per-user failed-login counts (as used in Fig. 3) also provide protection against sweatshop attacks and ATT relay attacks, especially such attacks targeting a particular account. Note that embedding warnings within ATTs (§7.1) does not by itself protect against sweatshop attacks.

For practical protection in Internet-scale live systems, we recommend combining techniques from §7 with those of §4. We see a large number of ways to expand on the ideas of §4. In particular, we encourage others to explore the use of dynamic parameters (ideally managed by automated tools), and other ways to gain advantage by treating users logging in from non-owned devices (e.g. traveling users) different from those continually using their regular login machines.

Acknowledgements: We thank anonymous referees for helpful comments. The first author acknowledges the support of the National Sciences Foundation under grant CCR0339464. The second author acknowledges the generous support of the National Sciences and Engineering Research Council of Canada for support as Canada Research Chair in Network and Software Security, and under an NSERC Discovery Grant.

References

1. M. Abadi, M. Burrows, M. Manasse, T. Wobber, "Moderately Hard, Memory-bound Functions", NDSS'03, San Diego, February 2003.
2. R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, 2001.
3. S. Bellovin, M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attack", *Proc. IEEE Symp. Research in Security and Privacy*, Oakland, May 1992.
4. S. Byers, A. Rubin, D. Kormann, "Defending Against an Internet-based Attack on the Physical World", Workshop on Privacy in the Electronic Society (WPES'02), November 21 2002, Washington D.C.
5. L. von Ahn, M. Blum, J. Langford, "Telling Humans and Computers Apart Automatically", *Communications of the ACM* vol.47 no.12 (Feb. 2004), pp.57-60. See also CAPTCHA Project web site (first appeared: 2000), http://www.captcha.net/
6. W. Diffie, M. Hellman, "New Directions in Cryptography", *IEEE Trans. Info. Theory* vol.22 (1976), pp.644-654.
7. W. Diffie, P.C. van Oorschot, M.J. Wiener, "Authentication and Authenticated Key Exchange", *Designs, Codes and Cryptography* vol.2 (1992), 107-125.
8. C. Dwork, M. Naor, "Pricing via Processing or Combatting Junk Mail", *Lecture Notes in Computer Science 740 (Proceedings of CRYPTO'92)*, 1993, pp. 137–147.
9. Password Usage, Federal Information Processing Standards Publication 112, U.S. Department of Commerce, NIST, 1985.
10. Automated Password Generator, FIPS Pub 112, U.S. Dept. Commerce, 1993.
11. W. Ford, B. Kaliski, "Server-Assisted Generation of a Strong Secret from a Password", 9th Int'l Workshop on Enabling Technologi (WET-ICE 2000), IEEE, 2000.
12. L. Gong, "Verifiable-text attacks in cryptographic protocols", *1990 IEEE INFOCOM*, pp.686-693.
13. L. Gong, T. Lomas, R. Needham, J. Saltzer, "Protecting poorly chosen secrets from guessing attacks", *IEEE J. Selected Areas Comm.* vol.11 (1993), pp.648-656.
14. D. Jablon, "Strong password-only authenticated key exchange", *ACM Computer Communcations Review*, Oct.1996.

15. A. Juels, J. Brainard, "Client puzzles: A cryptographic defense against connection depletion attacks", *Proceedings of the 1999 ISOC Network and Distributed System Security Symposium*, pp.151-165, 1999.
16. C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World*, Second Edition, Prentice Hall, 2002.
17. T. Lomas, L. Gong, J. Saltzer, R. Needham, "Reducing risks from poorly chosen keys", *Operating Systems Review* vol.13, pp.14-18 (presented at 1989 ACM Symp. on Operating Systems Principles).
18. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
19. M. Naor, "Verification of a human in the loop or Identification via the Turing Test", unpublished manuscript (1997), http://www.wisdom.weizmann.ac.il/∼naor/PAPERS/human.ps
20. B. Pinkas, T. Sander, "Securing Passwords Against Dictionary Attacks", *2002 ACM Conf. on Computer and Communications Security*, Wash. D.C.
21. A.M. Turing, "Computing Machinery and Intelligence", *Mind* vol.59 no.236 (1950), pp.433-460.
22. L. von Ahn, Eurocrypt'03 presentation of [23], 6 May 2003, Warsaw, Poland.
23. L. von Ahn, M. Blum, N. Hopper, J. Langford, "CAPTCHA: Using Hard AI Problems for Security", *Eurocrypt'03* proceedings, Springer-Verlag, LNCS 2656 (2003).
24. T. Wu, "The secure remote password protocol", Internet Society *1998 Network and Distributed System Security* symposium (NDSS'98).
25. T. Wolverton, Hackers find new way to bilk eBay users, CNET news.com 03/25/02.
26. J. Yan, "A Note on Proactive Password Checking", *Proc. 2001 ACM New Security Paradigms Workshop*, New Mexico, USA, Sept.2001.
27. J. Yan, A. Blackwell, R. Anderson, A. Grant, "The Memorability and Security of Passwords – Some Empirical Results", http://www.ftp.cl.cam.ac.uk/ftp/rja14/tr500.pdf, Tech. Report 500, Computer Lab, Cambridge, 2000.
28. P. Zimmermann, *The Official PGP User's Guide*, MIT Press, 1995.