Johnny Can Obfuscate; Beyond Mother's Maiden Name

William Cheswick
Lumeta Corp.

Abstract

Challenge/response authentication is stronger than password authentication, but has traditionally required a device for computing the challenge. Though human computation is limited, people can compute simple responses to challenges. If the challenge and the corresponding response is obfuscated with decoy information, an authentication scheme might be strong enough for a number of applications. The signs used in major league baseball provide some interesting techniques for obfuscation.

1 Introduction

Passwords have a long history of use, and of failure. They are subject to eavesdropping, guessing, and in many cases dictionary attacks. The human is the weak link in the chain: we would like him to choose cryptographically-strong keys, and he wants to remember something simple and get on with his work. Passwords can work acceptably if we can rule out eavesdropping and dictionary attacks. Unfortunately, eavesdropping is a common part of malware attacks.

Any one-time password solution is at least as good as, and usually better than fixed passwords. The strongest of these is challenge/response authentication[3], but that usually requires a hardware token, (expensive to deploy in quantity) or some sort of printout.[13] These something-you-have solutions are useless if you lose the token. A something-you-know solution would be more convenient, and could be worth deployment.

I propose to have the unaided human compute the proper response to a challenge. If successful, this technique would offer much of the strength of one-time passwords over fixed passwords, at much less cost and similar or greater usability. The user computes a simple response to a varying challenge, and is then encouraged to obfuscate this response in his answer. This can be fun, and a nice change from the strict spelling requirements of a password or passphrase.

A response of a single character might be sufficient for a rarely-used system. The hiding mechanism must be good enough that simple random authentication guesses will not succeed. We can reject the response if there is insufficient obfuscation. The number of unsuccessful attempts must be strictly limited, just as an automated teller will swallow or deactivate a card if too many incorrect PINs are entered. It would be nice if an eavesdropper watching a year's worth of daily logins would be unlikely to recover the algorithm used.

We can use a secret algorithm, or a public algorithm with some sort of key. Secret algorithms have a bad history in cryptography: we've learned that it is better to have a public algorithm, open for inspection, and a secret key. This paper presents a number of secret algorithms, but the goal is to reduce the problem to a key of some sort.

The motivation for this work is the demand for better authentication, especially in the banking and interactive community. We need better authentication than passwords, and better back-up authentication than personal questions like "mother's maiden name (MMN)".

The next section briefly touches the related work. Section 3 covers some of the basics of major league baseball signals, which offers some time-tested techniques for obfuscation. Section 4 describes two experiments with human computation. Section 5 examines the problem further, finishing with some open questions in Section 6.

2 Related Work

There is a rich literature on authentication, passwords, and one-time passwords, which will not be reviewed here. There are also a growing number of graphical challenge/response solutions: text-based solutions are more general, and can be implemented in Pluggable Authentication Modules (PAM)[15] on Unix systems.

Related work falls under several categories: *pass-algorithms* and *reconstructed passwords*,[4] *zero knowledge authentication*, *human-computer cryptography*,[10] *HumanAut*,[6] *Secure Human-Computer Identification (secHCI)*,[8] *cognitive trapdoor games*,[14] and *human interactive proofs* (HIP).[1] I will use *pass-algorithms*, the earliest term.

Text-based pass-algorithms fall into two categories, *ad hoc* and information-theoretic. The former are the oldest, and also the kind proposed in this paper. They start with proposals in Hoffman[5], Haskett[4], and Lipton[10], who gave a taxonomy of pass-algorithms.

In 1991, Matsumoto and Imai[12] created and analyzed mathematically a pass-algorithm, the first in the information-theoretic approach. The goal is a mathematical analysis that lays out the exact security against particular attacks for a given key length in their algorithm. This work has advanced with work by Wang *et. al.*[18], Matsumoto again[11], Li and Teng[9], Hopper[6], and Li and Shaum[8]. (The last has a good review of the literature, and in particular the theoretic approach.)

All of the information-theoretic approaches wrestle with the tradeoff between a sufficiently rich key for strong authentication, and usability. Keys must be short enough to be memorable and usable. I don't think they are there yet, but they may have reasonably simple solutions if they accept the weakened requirements suggested here. Roth *et. al.*[14] in particular seeks to obfuscate simple PINs in ways that seem more usable.

The field of military communications may offer other examples, or lessons, for our pursuit.

It is also possible that espionage fieldcraft may have some techniques of interest. I have not pursued either of these.

One technique of accessing a networked computer is *port knocking*.[7] A series of TCP connection attempts to certain ports, or special packets, can unlock a network server.

3 Baseball Signs

In a typical major league baseball game, nearly a thousand signs may be issued. Most of these signs are missed by the fans,[2] but they are an important part of the game, as are stealing signs from the other team.

The various coaches and players have to communicate some simple secrets in full public view. In particular, the pitcher and catcher negotiate what the next pitch is going to be, to help the catcher to prepare to catch it. The base coaches relay a steady stream of instructions to the batter and base runners from the head coach. Coordination of offensive plays can make the difference in a close game. In all cases, this information must be transmitted quickly and accurately, in public, to and by players who are under stress, and may not have graduated in the top of their class.

They provide some technologies that may help us confound eavesdroppers of our challenge/response transactions.

3.1 Pitcher/catcher communication

A fast ball travels at more than 90 MPH (144 kph), traveling from the pitcher to the catcher in some 300 ms. Neither the catcher nor the batter have much time to react to the pitch. The pitcher and catcher (the *battery*)[17] negotiate each pitch using a *finger system*:

1 fast ball
2 curve ball
3 change-up or other breaking ball

(There are less common signs such as "hit the batter" or "throw a pitchout.")

In the simple case, the catcher flashes a number of fingers, and the pitcher either accepts or "shakes off" the pitch, calling for another. But if there is an opposing player on second base, he can steal the catcher's signs and signal the pitch to the batter. Other have stolen signs using binoculars and signals from the scoreboard, etc.

To obfuscate his signs, the catcher may use several schemes:

```
ignore the first n signs
```

where n may change every few innings. Or

```
sum the signs until the sum is n
use the next sign
```

Another:

```
ignore all signs until the indicator sign
use the next sign
```

or a system used by Yogi Berra:

```
(compute the sign as above)
add a digit from the scoreboard
```

Or perhaps:

```
add all the signs
even sum: fast ball, odd: curve ball
```

Pitcher Whitey Ford used a custom pass-algorith:

```
count the number of signs shacken off
```

The signals were hidden in plain sight. This is common in baseball: often the character performing all the motions on the field is not giving any signs at all. Particular batteries often use their own specific algorithms. The history of pass-algorithms suggest that a number of programmers have done the same.

3.2 Signs from the third or first base coach

The manager usually relays offensive instructions through the base coaches, who are in plain sight on the field. The coach may transmit one of some 25 plays at each pitch. They are transmitted by a series of taps on various parts of his body, where he stands, position of his hat, etc. The signals are transmitted quickly: I have seen a coach make over 20 distinct motions in a few seconds, almost too fast to count. Missed signs are much more common than stolen signs.

The signs and their meanings can change game by game and even inning by inning. The opposition may have a former teammate, so the signs have to change when the roster does. The key of course, is that most of the signs are meaningless. The coaches use an *indicator sign* that says that the next sign is the *hot sign*. For example, the sign for a bunt might be touching the belt buckle, but only immediately after the indicator sign, which might be touching the nose. The rest of the signals are decoys, (*dekes*).

Here are some terms from Dickson:[2]

- **Activator:** Proceed with the play: a green light.
- **Automatic switch:** By default, reverse the meaning of a sign or signs until further notice. Not to be confused with a **flip**, a sign that reverses the meaning of the next sign only. The *battery* treats fast ball and curve ball as a binary state in this case.
- **Combination Signal:** two or more signs that combine to mean something. These are more likely to be missed.
- **Dead zone:** A sign that nullifies the next sign.
- **Live sign:** The sign conveying the actual message.
- **Indicator:** The signal that the live sign is coming. It may be the sign right before the live sign, or something more general, like the coach's position in the field.
- **Key:** The sign that unlocks the indicator.
- **Pump system:** Number of signals given is the signal, not the signals themselves.
- **Release sign:** End of message indicator.
- **Rub off** or **wipe-off** sign: Cancels every sign received so far.

Major league teams have gone to great lengths to steal signs. They may hide people with binoculars in the scoreboard or even neighboring buildings. The signs can be relayed back through buzzers or even subtle changes to outfield advertising.

> "If you want to steal signals the most important thing is to find the source of the signals" – Branch Rickey[2, p. 14]

The position of the tongue depressor scraping a shoe, or the last name of a player standing in the dugout, may be the actual sign, not the gesturing third base coach. This kind of indirection can be useful to us.

Important games have been lost and won through stolen signs. A coach who is good at sign-stealing can prolonge his career by a decade or more.

4 Experiments

4.1 Unlocking a queue manager

I tried a pass-algorithm in print queue software in a student PC lab in the mid-1980s. The privileged queue management commands were unlocked by the student running the lab after looking at the header on the screen:

```
Printer 3.   4 Nov 1985  10:49
```

He would do the following computation in his head:

```
take the minute's ten's digit
add the minute's units digit
add one
modulo 10
```

User	Challenge	Response	Correct?
ches	00319 Thu Dec 20 15:32:22 2001	23456bcd;f.k	y
root	00294 Fri Dec 21 16:47:39 2001	nj3kdi2jh3yd6fh:/	y
ches	00311 Fri Dec 21 16:48:50 2001	/ldh3g7fgl	y
ches	00360 Thu Jan 3 12:52:29 2002	jdi38kfj934hdy;dkf7	y
ches	00416 Fri Jan 4 09:02:02 2002	jf/l3kf.l2cxn.	y
ches	00301 Fri Jan 4 13:29:12 2002	j2mdjudurut2jdnch2hdtg3kdjf;s'/s	y
ches	00301 Fri Jan 4 13:29:30 2002	j2mdgfj./m3hd'k4hfz	y
ches	00308 Tue Jan 8 09:35:26 2002	/l6k3jdq,	y
ches	84588 Thu Jan 10 09:24:18 2002	jf010fk;.j	y
ches	84588 Thu Jan 10 09:24:35 2002	heu212jdg431j/	y
ches	00306 Thu Jan 17 10:46:00 2002	jfg.bv,vj/,1	y
ches	00309 Fri Jan 18 09:37:09 2002	no way 1 way is best!/1	y
ches	00309 Fri Jan 18 09:37:36 2002	jzw	no
ches	00368 Tue Jan 22 09:51:41 2002	84137405jgf/	y
ches	77074 Tue Feb 19 09:02:52 2002	d	no
ches	77074 Tue Feb 19 09:02:57 2002	hbcg3]'d/	y
ches	00163 Mon Feb 25 09:24:30 2002	d	no
ches	00163 Mon Feb 25 09:24:35 2002	ozhdkf0ey2k/.,vk0l	y
ches	00156 Tue Mar 12 12:41:12 2002	3+4=7 but not 10 or 4/2	y
ches	00161 Fri Mar 15 09:41:20 2002	/.,kl9djfir	y
ches	00161 Fri Mar 15 09:41:36 2002	3	no
ches	00160 Mon Mar 25 08:52:59 2002	222	y
ches	00160 Mon Mar 25 08:53:09 2002	2272645	y
ches	29709 Mon Apr 1 11:36:34 2002	4	y
ches	41424 Mon Apr 8 09:49:09 2002	ab3kdhf	y
ches	85039 Tue Apr 9 09:46:06 2002	04	y
ches	00161 Thu Apr 18 10:49:14 2002	898for/dklf7d	y

Figure 1: a *stockpile* of some selected challenges, responses, and whether they were accepted. The full list is available at `http://www.cheswick.com/ches/projects/data/pwlist`.

In this example, the response would be $((4 + 9 + 1) \bmod 10$ or "4". He typed in any three characters followed by this response digit.

I have no usability studies to offer for this algorithm. It seemed to work well, and I received no complaints about it. I am surprised I can still remember this algorithm after more than twenty years.

4.2 Login Authentication Test

I implemented a simple human-computed challenge/response PAM authentication module and used it as an additional login step for several months. Would it be annoying at times? Would I continue to use it over time. And how well would I bother to obfuscate my answers?

Most of the challenges and responses were captured to a file; Figure 1 shows a sample. Can you figure out the algorithm? The authentication code is shown in Figure2. It was occasionally annoying, and the backdoor authentication entry (a "/") was available. I regret that I didn't log its use.

It is not easy to figure out why each response is correct or incorrect. These could be charming, or maddening puzzles for the eavesdropper who collects enough samples. Automatic analysis seems difficult, but might not be.

5 Discussion

The human has two jobs: determine the correct answer from a given challenge, and then obfuscate that answer when typing it in. Indicator and wipe-off signs can be hidden in the challenge, combined with other memorized characters. Similar signs can be used in the responses.

The authentication algorithm in Section 4.2 was written in C. A suitable challenge/response language could be implemented as a little language.[16] I have tried to capture the flavor of such a language in the samples in this paper. Ultimately, the instructions must be easy for a layman to follow.

This little language could contain a number of primitives, a few of which are added together (perhaps at random) to make the password algorithm. Elements might include:

```
/*
 * A simple algorithmic password. Use
 * sparingly, and don't let anyone see
 * this code.  A job for obfuscated C?
 */
int
ok(const char *chal, const char *resp) {
    const char *cp;
    char *ep;

    if (resp[0] == '/') /* joshua */
        return 1;
    cp = strpbrk(resp, "0123456789");
    if (cp)
        ep = strpbrk(cp+1, "0123456789");
    else    return 0;
    if (ep) {
        cp = ep;
        ep = strpbrk(cp+1, "0123456789");
    }
    if (ep) cp = ep;
    return ((chal[3] - '0' + chal[23] -
        '0' + 1) % 10) == (*cp - '0');
}
```

Figure 2: The code used to validate the responses in Figure 1

```
find a word
skip to the next word if present
select word
count the consonants
count the characters
count the vowels
add n
subtract n
modulo n
remainder n
set indicator
find indicator
set wipeoff character
skip to wipeoff character
find a digit
find next digit, if any
select digit
```

The language might include stuttering: read a digit, and then skip that many characters and continue.

The keyboard can help map digits to a character. For example, on the QWERTY keyboard, the digit 6 could refer to the keys `6`, `y`, `h`, and `n`; another digit could select one of these. Adding or subtracting a row or column is easy to do, and increases the obfuscation.

An authenticator that occasionally rejects valid responses may also frustrate attempts to crack the pass-algorithm.

It is hard to remember passwords that are only used occasionally, say, a couple times a year. An pass-algorithm might be even harder to remember. If deployed widely, customer support costs will be a factor, which is one of the reasons MMN backup authentication has been implemented.

It may be that algorithmic passwords are not for everyone: Babe Ruth was known for forgetting signs in a matter of seconds. They might be popular enough to be a user-selectable alternative to other authentication schemes. Young users might enjoy the puzzle and obfuscation aspects of the algorithm.

Since users generally use the same password, possibly with some site-specific modification, it might be easier to use the same algorithm for several sites, but with a site-varying indicator or wipe-off symbol. This cross-site similarity is a weakness, but certainly less weak than using the same fixed password.

6 Open Questions

1. Using obfuscation, can we generate pass-algorithms with better usability and memorability characteristics than the information-theoretic approaches, and can the results be acceptably secure for important applications?

2. Are brute force algorithmic searches a feasible way to crack these simpler pass-algorithms?

3. What percentage of users would find pass-algorithms usable, or even fun to use?

4. Is it feasible to implement a single algorithm with different indicators wipe-off signs, etc., on different web pages? This could solve the single password on multiple web accounts if an authentication service were available to a large number of web developers.

5. Does user-generated obfuscation make the job of an occasional eavesdropper sufficiently hard?

6. Should pass-algorithms be offered as either an optional alternative, or optional addition, to normal authentication?

7. Should people be allowed to supply their own pass-algorithms?

8. If people supply their own pass-algorithm code, what kind of automated tests can we apply to enforce some reasonable security.

9. Should we record their authentication history (as in Figure 1) to allow auditing by security officers. Obviously, this would be a dangerous log in the hands of the wrong people.

The most useful outcome of these explorations would be a better authentication technology for public use. It is unclear whether Joe Sixpack (favorite PIN: 1234) could handle this kind of authentication, or remember the specifics over a period of months. But perhaps it could be used by those who opt-in for it. And couching the approach in baseball terms might help in the United States—I have heard of teachers using batting averages to teach percentages and probability.

It is unclear to me how to put all these pieces together into a either a public algorithm with personal indicators and wipe-off characters, or mass-produced customized algorithms. Any proposals would need usability studies.

7 Acknowledgments

My thanks to Mike Reiter, Dave Wagner, Sam Baskinger, Dan Boneh, Bob Cousins, and Peter Honeyman.

References

[1] Chew, M., and Baird, H. Baffletext: A human interactive proof. In *Proceedings of SPIE-IS&T Electronic Imaging, Document Recognition and Retrieval X* (jan 2003), pp. 305–316.

[2] Dickson, P. *The Hidden Language of Baseball*. Walker & Co., 2003.

[3] Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO* (1986), pp. 186–194.

[4] Haskett, J. A. Pass-algorithms: A user validation scheme based on knowledge of secret algorithms. *Communications of the ACM 8*, 27 (1984), 777–781.

[5] Hoffman, L. J. Computers and privacy: A survey. *ACM Comput. Surv. 1*, 2 (1969), 85–103.

[6] Hopper, N. J., and Blum, M. Secure human identification protocols. *Lecture Notes in Computer Science 2248* (2001), 52–??

[7] Krzywinski, M. Port knocking: Network authentication across closed ports. 12–17.

[8] Li, S., and Shum, H.-Y. Secure human-computer identification (interface) systems against peeping attacks: Sechci. Cryptology ePrint Archive, Report 2005/268, 2005. `http://eprint.iacr.org/`.

[9] Li, X.-Y., and Teng, S.-H. Practical human-machine identification over insecure channels. *J. Comb. Optim. 3*, 4 (1999), 347–361.

[10] Lipton, D. Logical authentication method. *SIGSAC Rev. 4*, 2 (1986), 9–20.

[11] Matsumoto, T. Human-computer cryptography: An attempt. In *ACM Conference on Computer and Communications Security* (1996), pp. 68–75.

[12] Matsumoto, T., and Imai, H. Human identification through insecure channel. In *EUROCRYPT* (1991), pp. 409–421.

[13] Peters, B. Security considerations in a multiprogrammed computer system. In *Proc. AFIPS 1967 Spring Joint Comput. Conf.* (Washington, D.C., 1967), Thompson Book Co.

[14] Roth, V., Richter, K., and Freidinger, R. A pin-entry method resilient against shoulder surfing. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security* (New York, NY, USA, 2004), ACM Press, pp. 236–245.

[15] Samar, V. Unified login with pluggable authentication modules (pam). In *ACM Conference on Computer and Communications Security* (1996), pp. 1–10.

[16] sdalus, P. *HPL: Little Languages and Tools*. Macmillan Technical Pub, July 1998.

[17] Southworth, S. *The Complete Book of Baseball Signs and Plays*. Coaches Choice, 1999.

[18] Wang, C.-H., Hwang, T., and Tsai, J.-J. On the matsumoto and imai's human identification scheme. In *EUROCRYPT* (1995), pp. 382–392.