How to Speak an Authentication Secret Securely from an Eavesdropper

Lawrence O'Gorman, Lynne Brotman, and Michael Sammon

Avaya Labs, Basking Ridge NJ 07920, USA
{logorman,lynne,mjps}@avaya.com

Abstract. When authenticating over the telephone or mobile headphone, the user cannot always assure that no eavesdropper hears the password or authentication secret. We describe an eavesdropper-resistant, challenge-response authentication scheme for spoken authentication where an attacker can hear the user's voiced responses. This scheme entails the user to memorize a small number of plaintext-ciphertext pairs. At authentication, these are challenged in random order and interspersed with camouflage elements. It is shown that the response can be made to appear random so that no information on the memorized secret can be learned by eavesdroppers. We describe the method along with parameter value tradeoffs of security strength, authentication time, and memory effort. This scheme was designed for user authentication of wireless headsets used for hands-free communication by healthcare staff at a hospital.

1 Introduction

When we type a password into a computer or a PIN into a bank machine, the characters are usually masked on the screen (*e.g.*, "******") to prevent onlookers from seeing the secret code. When speaking a password or PIN into a telephone or other voice communication device, there is nothing comparable to keep the password secret. There are two alternatives for the user. One is to speak softly and to hope no eavesdropper hears. Another is to use a telephone keypad and to make sure that no onlooker sees which keys are pressed. Neither solution is very satisfactory. The former depends on the user finding a place out of earshot from others. The latter precludes hands-free communications and fails to offer a solution for present and future communications devices that use only voice communications. The objective of this work is to offer a means for user-authentication by voice in which an eavesdropper who can hear the user responses cannot gain information to impersonate the true user.

We propose a method called SPIN, for "spoken PIN". This authentication protocol involves the user first memorizing simple plaintext-ciphertext pairs, *e.g.*, red=3, green=2, blue=9. A plaintext sequence is sent – spoken – by the authentication server to the user. We assume the user receives this sequence by an earphone or in some way that eavesdroppers cannot hear it. The user responds with the ciphertext element associated with each plaintext element. We assume

B. Christianson et al. (Eds.): Security Protocols 2006, LNCS 5087, pp. 215–229, 2009.

there may be eavesdroppers hearing this response. We show that by randomizing the sequence of authentication elements and by interspersing *camouflage* elements in the sequence, that an eavesdropper can obtain no information to learn the cipher over one or many responses.

In Section 2, we describe some alternative authentication methods where the user response can be seen or heard by eavesdroppers and the procedure can still be secure. In Section 3, we detail the SPIN method and provide equations and plots to choose among parameter value tradeoffs. We also make some security and convenience comparisons between SPIN and PINs, passwords, and one-time passwords. We describe our healthcare application of SPIN in Section 4 and discuss SPIN and some alternatives in Section 5.

2 Background

There is a common solution to any authentication problem where eavesdroppers may hear or see the user's response. This is not to repeat that response. Or, more specifically, the correct authentication response will change randomly each time it is given. This is different than a traditional, static password. One reason the changing authentication response protocol is less common than the static password protocol is that, for a changing response the user usually requires some sort of aid, such as an electronic token, to correctly respond with a different response for each authentication instance. One-time passwords, time-synchronous PINs, and challenge-response pass codes are all examples of this.

When one speaks of spoken authentication, one might picture a secret rendezvous between spies where they identify themselves by a pre-arranged dialogue such as,

Spy 1 – *"I hear Cyprus is lovely this time of year."*
Spy 2 – *"The leeward beaches are best in the afternoon sun."*

As much as a dialogue such as this might help the two spies identify each other, it can only be used once because an eavesdropper who has heard the dialogue can impersonate either spy subsequently. This is akin to a one-time password scheme of which there are computer authentication examples such as SKEY, which is a chained list of hashes that the user carries and uses one each time until the list is depleted [1, 2]. Because each password is used once, it would be tedious for most users to memorize the list, so instead it is carried with the user, either in paper form or on a portable electronic device.

Time-synchronous pass codes work in the following way [3]. A user has an electronic token that generates a different number periodically, say every minute. The server to which the user is authenticating generates the same number sequence synchronously to the token. Therefore, when the user wants to authenticate, she sends the current pass code to the server for confirmation that it is the current number. Even if an eavesdropper sees the number, this will not be useful in subsequent authentication attempts because the sequence changes randomly.

A challenge-response pass code is similar to the time-synchronous one in that a user must also carry an electronic device. In this case, the user first requests to authenticate to the server. The server sends a random number challenge. The user enters the random number into the device, which performs cryptographic calculations to generate a response pass code that is sent back to the server.

The Query-Directed Password (QDP) scheme [4, 5] is a challenge-response scheme using multiple-choice, personal questions, such as, "What was the color of the car on which you learned to drive? 1) black, 2) white, 3) blue, 4) red, 5) green, 6) gray." If the user responds with the number of the multiple-choice answer for each question, and if the questions and/or the numbers associated with answers are randomized between authentication instances, then an eavesdropper will hear a different sequence of numerical responses each instance. Implemented correctly, responses will appear to an eavesdropper as random numbers. QDP uses a number of personal questions, or challenge questions, that are more varied and secure than the common "mother's maiden name" challenge [6-8]. The user does not have to memorize anything because answers to the personal questions will already be known (if chosen well). However, there are some drawbacks to this approach. One is a time-security tradeoff. It takes time to read each question including multiple choices. Furthermore, more questions and/or more answer choices are required for higher levels of security (4-5 questions of 6 multiple choices each were used in our testing). We compare this QDP approach with the proposed method of this paper in Section 3.5.

Our voiced-password problem would seem perfectly suited for a speaker verification solution. However, for our application, which involved health care workers speaking over a headset in an often noisy hospital environment, the recognition rate (or verification rate) is not reliable enough at this time.

There is an analogous user authentication procedure that also involves a one-sided eavesdropper. This is a graphical password scheme that can defend against a Trojan keyboard logger (*e.g.*, [9]). A keyboard logger program that has been secretly installed on a victim's computer can record all keystrokes for an attacker to learn passwords. A graphical password scheme can defend against this in the following way. Pictures are displayed on the screen with identifying numbers, but these numbers change each authentication session. The user is asked to enter the numbers associated with the pictures she has memorized for authentication. Analogously to an eavesdropper hearing a spoken authentication response for our application, this logger can capture the authentication numbers that are entered, but because the numbers change each time, no useful authentication information is gained.

Passwords, challenge questions, and biometrics involve human factors such as memory and physiology, as well as security considerations. This combination – often a tradeoff – between human factors and security is also important to security protocols where the human is involved. Security protocols involving humans have been used since the Egyptian, Greek, and Roman empires (and before) [10]. New human security protocols are being proposed and analyzed with current-day knowledge of computer security [11]. SPIN is a user-authentication

scheme that falls into this category of human security protocols because a human performs her half of the protocol without aid of a computing device.

3 SPIN–Spoken PIN

The SPIN method is described first by example in Section 3.2. Although the method itself is straightforward, it is not so obvious how to optimize the security-time-memorization tradeoffs. Section 3.3 describes parameters of the method and we examine the tradeoffs in Section 3.4 with respect to these parameters. We begin with some definitions.

3.1 Definitions

The solution presented here to securely speak an authentication secret involves the use of a substitution cipher, a challenge-response protocol, and camouflage elements. We define these terms here.

A *substitution cipher* is a secret code where a sender substitutes characters of plaintext in a message by characters of ciphertext, and sends this coded message to the receiver. The receiver who knows the cipher inverts the substitutions to recover the plaintext. A simple substitution cipher might substitute each character by the character one above in the alphabet (the simplest form of Caesar cipher [10]). The secret describes the substitution rule(s) and only the sender and authorized receiver(s) should share this knowledge.

A *challenge-response protocol* used in user-authentication is an exchange between an authentication server and the authenticating user whereby the server sends a message to the user that changes each time, and the user returns a response that depends upon the challenge. A primary advantage of challenge-response protocols for user-authentication is that an attacker cannot simply replay the response, because it is different for each challenge. In this paper, we refer to the exchange between server and user as a *challenge-response sequence*, because there is not just one challenge and one response, but a sequence of these that make up a single authentication session.

An *element* is one component of a sequence. This element can be a character, word, or number. In this paper, challenge elements are colors, and response elements are numbers, but either could equally well be letters, words, animal names, names of planets, etc. An *authentication pair* consists of a single challenge element and its corresponding response element. An *authentication element* is a general term for either a challenge element or a response element, in other words it is just one of the elements that is used for authentication. In contrast, a *camouflage element* is one that is not used for authentication. It is interspersed with authentication elements to reduce the chance that an eavesdropper will learn the substitution rules after hearing one or more authentication sequences. In the next section we use the convention of bold font for authentication elements to distinguish these from camouflage elements in regular font.

We describe two major attackers. One is the *eavesdropper* who we will call *Eve*. She can hear user responses. We assume we have no control on Eve, so she

can hear an unlimited number of responses to try to gain authentication. The other attacker is *Brutus*, who can steal the headset in some brute force manner, then *hear authentication challenges* and try an exhaustive guessing approach. In addition, Brutus can learn from listening to repeated challenges, if given the chance to do so, to mount a more sophisticated attack. We have some control on Brutus because we know when he answers erroneously. Since each element is challenged and responded to individually, we have the ability to respond to erroneous elements by, for instance, freezing the account. So, Brutus's freedom to attack is much more limited than Eve's. These two attackers can collude.

The description of the two attackers above also defines the *threat model* that we examine in this paper. The SPIN threat model is alike that of passwords and PINs in most ways. Like these traditional authentication methods, a user's SPIN code must be memorized and kept secret from attackers. If lost, an attacker can gain access to the user's account. If forgotten, the user cannot authenticate. SPIN is different than passwords and PINs in one major way: there is no threat if the SPIN response is learned by attackers. Just like passwords and PINs, a more comprehensive SPIN threat model also includes issues such as security of the channel (in this case the wireless channel), protection of the secret at the server, etc. But, it is the threat from Eve and Brutus that we examine in depth here.

Finally, we define the term *security strength* as being the total number of attempts an attacker must make to be sure to find the authentication response: the more attempts required, the greater the security strength. For this paper, the security strength is generally the number of permutations an authentication challenge can take. (Note that a brute force guesser will likely guess the answer in half the number of permutations on average.)

3.2 Description by Design Progression and Example

The SPIN method is a straightforward substitution cipher into which we intersperse camouflage elements. We describe it in this section by a progression of methods, starting with the most straightforward and adding modifications to address shortcomings, finally leading to the proposed method. In the following section, we generalize the method.

Method a – First, consider a simple substitution cipher where colors are replaced by numbers. For instance,

Substitution rules: **3** → **Red**, **2** → **Green**, **9** → **Blue**, **6** → **Yellow**.

An authentication session using the rules above might look like,

Challenge from server: **Blue, Red, Yellow, Green**
Response from user: **9, 3, 6, 2**
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

Since the challenge changes each time, Eve cannot merely hear **"9, 3, 6, 2"** and replay it to successfully authenticate. So, we have made one step toward the goal of speaking a password without an eavesdropper being able to decipher and repeat it.

However, there is a problem with this simple cipher method. If Eve hears a few responses, even though the elements will be in different order, she will soon learn that the cipher code only uses elements whose numbers are **{2, 3, 6, 9}**. Therefore the number of different permutations that these digits might take, is reduced from $10 \times 9 \times 8 \times 7 = 5040$ for any 4 different digits randomly ordered, to $4 \times 3 \times 2 \times 1 = 24$ for these 4 specific digits.

Method b – To strengthen the security, the server can randomly intersperse camouflage elements in the challenge sequence. Using the same substitution rules as above, an example of an authentication session is,

Challenge from server: 1, 8, **Blue**, 4, **Red**, **Yellow**, 0, 7, **Green**, 5
Response from user: 1, 8, **9**, 4, **3**, **6**, 0, 7, **2**, 5
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

The user performs the substitutions only for the colors and simply repeats the camouflage elements. The server needs only to check that the substitutions are correct and that the camouflage elements have been repeated. The camouflage is present only to prevent Eve from learning the authentication elements in the response. Since, as can be seen in the example, there are 10 different digits in the response, Eve will always hear some different ordering of 10 digits and will not learn anything about the cipher.

Although Eve does not gain information from these permutations of digits 0-9, Brutus can gain information from hearing the challenge sequence. After hearing an entire sequence, Brutus knows that the color substitution values are all the numbers that were not present in the challenge. So, this again reduces the permutations from $10 \times 9 \times 8 \times 7 = 5040$ to $4 \times 3 \times 2 \times 1 = 24$.

Method c – Instead of inserting camouflage elements that are dependent upon the authentication elements, such that all digits from 0-9 are present exactly once in a sequence, let's try choosing camouflage elements randomly. The larger the number of camouflage elements we choose, the greater is the chance that one or more will overlap with the values of the authentication elements. When this happens, Brutus, who hears a challenge, will not be able to know all the color substitutions just by hearing the numbers not present in the challenge. In the following example, we've added 6 random camouflage elements,

Challenge from server: 1, 8, **Blue**, 9, **Red**, **Yellow**, 1, 7, **Green**, 3
Response from user: 1, 8, **9**, 9, **3**, **6**, 1, 7, **2**, 3
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

The color substitution values for blue and red happen to appear in the camouflage elements. Therefore, Brutus who depends on learning authentication elements by numbers that are not present in the sequence, will not learn that 9 and 3 are color substitution values. If we increase the number of this type of camouflage elements (to infinity), we increase the assurance that all color substitution values will be present so the attacker's information decreases (to zero).

However, by adding camouflage elements independent of authentication elements, we can succumb to an attack at the eavesdropper's end. If Eve listens

to a few response sequences, she can learn that authentication elements, which are always there, occur with higher frequency than camouflage elements, which don't have to be there. We call this a histogram attack. Once Eve has heard enough responses to learn the authentication elements, this again degrades to an attacker needing only $4 \times 3 \times 2 \times 1 = 24$ guesses.

Method d – Since Methods b and c defend against Eve or Brutus separately, perhaps we can combine these approaches to yield method resistant to each. Start with authentication elements randomly ordered. Randomly insert camouflage elements chosen dependently as described in Method b. Then, randomly insert more camouflage elements independently as described in Method c. In the following, we've added 3 independently chosen camouflage elements to the 6 dependently chosen camouflage elements already there and the 4 authentication elements,

Challenge from server: 1, 8, **Blue**, 4, 9, **Red**, **Yellow**, 0, 1, 7, **Green**, 3, 5
Response from user: 1, 8, **9**, 4, 9, **3**, **6**, 0, 1, 7, **2**, 3, 5
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

Eve obtains no information from hearing the response sequence because she still hears a permutation of 0-9, but now there are added digits chosen independently of authentication elements, so this gives no additional information.

Now, Brutus who listens to the challenge will hear every digit except 2 and 6. So he knows that two color substitution values must be these digits, but has no information on the other two. If we increased the number of independently chosen camouflage elements, we would eventually include all color values (with some probability) such as to defend against Brutus (with some probability).

Method e – Instead of trying to hide the authentication elements in a probabilistic fashion as in Method d, we can do this deterministically by slightly modifying the way the challenge sequence is given. Instead of a challenge element being "number" or "color", each is, "number or color". When the color is not an authentication element, the user just ignores it and repeats the number. When the color is an authentication element, the user ignores the number and responds instead with the authentication number corresponding to the color. Following is an example of this combination,

Challenge from server: 1 or Purple, 8 or Black, 3 or **Blue**, 4 or Pink,
2 or **Red**, 6 or **Yellow**, 0 or Orange,
7 or Gray, 9 or **Green**, 5 or White
Response from user: 1, 8, **9**, 4, **3**, **6**, 0, 7, **2**, 5
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

From this example, there are two types of challenge elements. One type is the camouflage element (*e.g.*, "1 or Purple") that contains a number and color, and neither the number nor color can be an authentication element. There is no correspondence between these camouflage colors and numbers, their pairings are random. The second type of challenge element is the authentication element (*e.g.*, "3 or **Blue**"). This contains a number and an authentication color. The

number must be one of the authentication number responses, but there is no correspondence between an authentication number and color in a single element; indeed as can be seen in the example, "6 or **Yellow**", the number and color can be the correct authentication pair. No color or number can repeat in a challenge sequence.

Now, Brutus who listens to the challenge will hear all the possible digits and all the possible colors. So, he will not be able to ascertain which digits are authentication elements by their absence in the challenge sequence (as in Method c). Furthermore, Eve will always hear a randomly ordered sequence of every possible response repeated once, and once only. Therefore, she will not gain any information over an unlimited number of responses.

So, at the expense of a little more time to speak a more wordy challenge sequence, we have defended equally against both Brutus and Eve for a single authentication challenge-response. There is one small addition we make to this protocol. We said that the numbers spoken for authentication elements in the challenge sequence are random orderings of authentication numbers. This is true for any ordering except for the one where authentication numbers and colors all happen to correspond exactly, for which we make an exception. For the example above, this is a challenge sequence containing any ordering of all the pairings, "**9** or **Blue**", "**3** or **Red**", "**6** or **Yellow**", "**2** or **Green**". Although this random pairing has the same probability as any other, we do not use it as a precaution against the "naïve" attacker who just repeats all the challenge numbers and would then succeed to authenticate for this single case.

From this section, we have found two methods that offer security against both Eve and Brutus. Method d offers a probabilistic measure of security at the expense of additional camouflage elements. Method 3 offers a deterministic measure of security at the expense of more complexity in the protocol. In the next section, we generalize the approach and in Section 3.4 examine tradeoffs in security, time to authenticate, and memorization effort.

3.3 Formal Description

In general, the SPIN code can be described by the following parameters,

$$\textbf{SPIN} \quad (m, a, c_D, c_I, L) \tag{1}$$

m = number of memorized substitution pairs

a = number of authentication elements in a code

c_D = number of dependent camouflage elements in a code

c_I = number of independent camouflage elements in a code

L = number of levels, or values, that elements can take.

From Method b, the minimum number of elements in an authentication sequence is $n_{min} = a + c_D$. Because the c_D elements are chosen to be all the element values that are not authentication element values, then $c_D = L - a$, so $n_{min} = L$. From

Method d, we insert independent camouflage elements to the sequence, so the total number of elements in a sequence is,

$$\text{Total number of elements: } n = a + c_D + c_I = L + c_I. \tag{2}$$

From Section 3.2, the best- and worst-case security strengths are,

$$\text{Best case security: } S = L(L-1)(L-2)\ldots(L-a+1) = \binom{L}{a} a! \tag{3}$$

$$\text{Worst case security: } S = a!. \tag{4}$$

For Method e, there are no independent camouflage elements. So, $c_I = 0$ in equation (2) and the total number of elements in a sequence is, $n = a + c_D = L$. Because this method defends equally against Eve and Brutus, the security strength is the same for both and is the best-case security strength, equation (3), minus 1 due to the exception of excluding the case where challenge numbers and colors correspond, as mentioned above.

Method d is less straightforward than Method e because of the independent camouflage elements and the fact that security strength is affected by these elements probabilistically. The best-case security strength for Method d occurs for the case when all authentication elements are located at the beginning of the authentication challenge. In this case, Brutus obtains no information from the camouflage elements, because he has to make his guesses before learning the information he gains from hearing them. An example of a best-case challenge is, "**Blue, Red, Yellow, Green**, 0, 4, 7, 1, 8, 5".

The worst-case security strength occurs for the case when all authentication elements are located at the end of the authentication challenge. In this case, Brutus has obtained as much information as there is from the camouflage elements before having to guess the authentication elements. An example of a worst-case challenge is, "0, 4, 7, 1, 8, 5, **Blue, Red, Yellow, Green**".

When we add c_I elements, we can raise the worst-case security strength closer toward best-case. This is because some c_I elements might have the same values as authentication elements, thus preventing Brutus from learning about authentication element values through their absence. However, because the c_I elements are chosen independently of authentication element values, we can only say probabilistically whether there will be repeating elements. Obviously, the more the c_I elements there are, the greater the chance of repeating authentication elements. To understand the effect of adding c_I elements, we need to determine the number of elements, c_I, necessary to obtain a probability P that k or more of the a authentication elements ($k \leq a$) is repeated in the c_I elements.

The probability that $k = a$ authentication elements are present in c_I randomly chosen elements can be derived as follows:

$$P(k = a = 1, c_I) = 1 - ((L-1)/L)^{c_I}$$
$$P(k = a = 2, c_I) = 1 - 2\left((L-1)/L\right)^{c_I} + \left((L-2)/L\right)^{c_I}$$
$$\ldots$$

This can be described by the forward difference operator, $D(f(x)) = f(x+1) - f(x)$,

$$P(k, c_I) = D^{c_I}\left((L-k)/L\right)^{c_I}$$

In general, the probability is,

$$P(k, c_I) = \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} \left((L-k+i)/L\right)^{c_I} . \tag{5}$$

Besides the probability that *all* authentication elements are repeated in the independent camouflage elements, we will also ask the probability that *all or one less than all* authentication elements are repeated in the independent camouflage elements. This is,

$$P(k = a \text{ or } k = a-1, c_I) = P(k = a, c_I) + a\left(P(k = a-1, c_I) - P(k = a, c_I)\right) . \tag{6}$$

3.4 Security-Time-Memorization Tradeoffs

In practice, we want to optimize with respect to three parameter values. We desire high security strength, short authentication sequence (or session) length, and a small number of authentication pairs that a user has to memorize.

First, we simplify our choices by setting $m = a$. This implies that all the user's memorized elements are used in each authentication session. Although this does not have to be the case and there are security advantages of memorizing more pairs than are used each session, we do not explore this here, choosing instead to minimize the memorization effort of the user.

Secondly, for Method d, we make a choice for the probability value that affects the number of independent camouflage elements in Method d. We choose this to be, $P(k, c_I) = 66.6\%$. This choice is somewhat arbitrary, but we feel it to be practical at least for our own application described in Section 4. In words, this means that we are calculating the number of independent camouflage elements in which there is a two-thirds probability that a certain portion of authentication elements are repeated in these. We make another choice that this "certain portion" we calculate for will be all authentication elements, a, or all or one less than all authentication elements, a or $a-1$.

With equations (2-6), we can plot the security-time-memorization tradeoffs for Methods d and e. Figure 1 is for Method d for 66% probability that all or one less than all authentication elements will be in the independent camouflage elements. Figure 2 is for Method e.

We can obtain an idea of how these methods compare by examining a couple examples. For a requirement of $S = 1000$ and $m = 5$, we need about $n = 20$ for Method d in Figure 1, but only about $n = 6$ for Method e in Figure 2. For a requirement of $S = 100$ and $m = 4$, we need about $n = 15$ for Method d in Figure 1, but only about $n = 5$ for Method e in Figure 2.

Although we don't plot Method d for the 66% probability case that all authentication elements are in the c_I camouflage elements, we can obtain a comparison

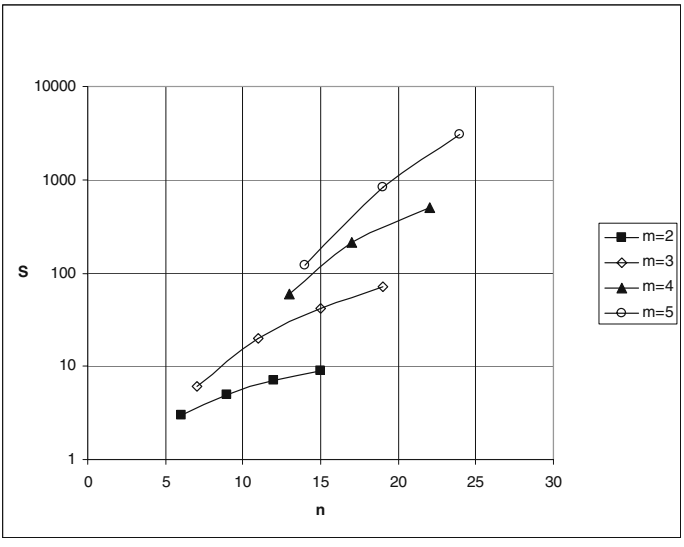

Fig. 1. Plot for Method d for 66% probability that all or all minus one authentication elements will be in the cI camouflage elements. Plot shows tradeoff among number of authentication elements the user must memorize, m, the security strength, S, and the length of the authentication sequence, n. The points are calculated for $L = \{4, 6, 8, 10\}$ on $m = 2$ and $m = 3$, $L = \{5, 6, 8, 10\}$ on $m = 4$, and $L = \{6, 8, 10\}$ on $m = 5$ (all from low to high n).

from the equations. For a requirement of $m = 4$ and $S = 1000$, Method d can attain this with $n = 24$. In Figure 2, Method e requires a far shorter sequence, $n = 7$. For a requirement of $S = 10,000$ and $m = 5$, we need about $n = 30$ for Method d, but only about $n = 9$ for Method e.

In these examples, the required sequence length was greater than 3 times for Method d than Method e. Although Method e is has a wordier challenge sequence, it takes only about 50% more time per challenge-response element for Method e than for Method d. Therefore, based on length of sequence (or time to authenticate) alone, Method d takes about twice as long, therefore is less desirable from this respect than Method e.

3.5 Comparing SPIN to PINs, Passwords, One-Time Passwords, and Challenge Questions

From Section 3.1, the security strength of a randomly-generated authentication response is the number of different possibilities it may take. For a PIN with 4 digits, the security strength is 10^4. For a Method e SPIN code we can achieve $S = 10^4$ with 5 authentication elements and a sequence length of 8, or with 4 authentication elements, we can achieve less security of 5040 with a sequence length of 10. In this example and in general, SPIN requires more memorization

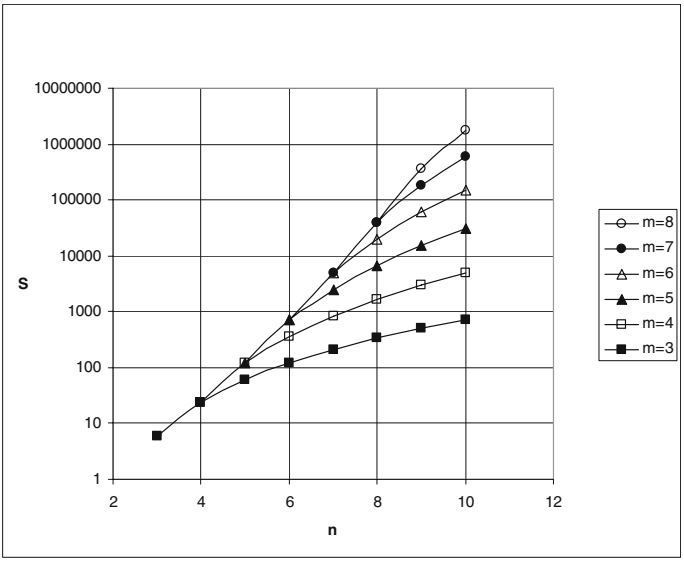

Fig. 2. Plot for Method e. Plot shows tradeoff among number of authentication elements the user must memorize, m, the security strength, S, and the length of the authentication sequence, $n = L$.

effort or a longer time to authenticate, or both, to achieve a similar level of security.

A password achieves much stronger security than a comparable-length PIN because there are many more choices of characters. For an 8-character password made up of any combination of upper and lower-case characters and digits, security strength is 62^8. One might initially suspect that we can gain such efficiency with SPIN as well by having many more levels (*i.e.*, more colors in our examples above) such as to increase the security strength. It is true this will increase the security strength, but there is a huge cost, as seen in equation (2), $n = L + c_I$. The total authentication sequence length must be equal to or larger than the number of levels of authentication elements. Even if only lower-case characters were considered, a 26-element authentication sequence would require too lengthy a response from the user.

As mentioned in Section 2, a one-time password from a list or token that the user carries will meet the needs of resisting an eavesdropper. Since a one-time password can contain any combination of characters and digits, it can have strong security per number of characters spoken. Therefore, this is a good alternative to SPIN — except for cases where we don't want to burden the user to carry a token or where we desire hands-free authentication.

QDP or challenge questions [4, 5], where the user responds with yes/no or multiple choice answers, can meet our specification for secure voiced authentication. The advantage is that these questions can be designed to be more easily remembered by the user. The disadvantage is the time required to achieve a

reasonable level of security. For QDP, if 5 questions are asked with 6 multiple choices each, this takes about $5 \times 15 = 75$ seconds and yields a security strength of $6^5 = 7776$. A comparable Method e SPIN code is for $m = 5$, $n = 8$, $S = 6720$. At 3 seconds per challenge-response element, this takes about 24 seconds, less than one third the time of QDP.

It is clear that SPIN fares less well against password, PIN, and one-time password for each comparison in this section of security strength, sequence length and memory effort. However, because none of the traditional schemes meets the requirement of hands-free, spoken authentication, these disadvantages are the price to achieve this. The main tradeoff between SPIN and QDP is memory effort versus time to authenticate.

4 Use of SPIN in a Health–Care Application

The SPIN code was initially developed to meet a need of MACCS (Mobile Access to Converged Communications Service), a system initially targeted for use by healthcare workers. This is a wireless, voice-interactive, hands-free communications system. Each user has a wireless headset for issuing voice commands to the system and communicating with other users. To protect users' and patients' private information that is communicated on MACCS, the users must authenticate themselves to the system. A static password or PIN is not acceptable because it can be overheard by eavesdroppers, and a one-time password requiring a token or list is unacceptable because it is not hands-free. QDP (as described in Section 2) was the initial authentication scheme used in MACCS, however the 3-4 questions took over 1 minute per authentication session. To reduce authentication time, SPIN was provided as an alternative to QDP.

Initial response to the use of SPIN was mixed. As compared to the QDP method, users were happy to authenticate more quickly. However, they were not happy to memorize another security code. We will report on our experiments and their results in a later paper.

5 Summary and Discussion

In this paper, we have described two variants of a method for speaking a password securely, potentially in front of eavesdroppers. The method involves the user memorizing color-number pairs. A challenge is sent to the user involving camouflage numbers and authentication colors. The user repeats the camouflage numbers and substitutes colors for the corresponding, memorized numbers. We have described a protocol whereby this authentication scheme can be secure from eavesdroppers and brute force attackers. There seems to be little or no treatment in past literature on this topic. However, while working and testing QDP and SPIN, other spoken authentication methods have arisen. We have not performed user testing with these, so we merely mention these alternatives as potential for future work.

One method decreases the time (number of elements) by requiring the user to perform some elementary mathematics. The user memorizes m digits. A challenge consists of m randomly chosen digits. The user responds with the addition of each challenge digit to a memorized digit, modulo-10. This eliminates the need for camouflage elements because the response that Eve hears will be random.

Another method reduces the memory effort. Instead of using random color-number pairs, more memorable pairing schemes can be used. Colors can be listed by the user's order of preference, or country names can be used by the user's order of preference for vacations, etc.

The QDP method can been enhanced to perform a variant of the method just mentioned. For instance, a full QDP question might be, "Where was the family car parked in relationship to your childhood home? 1) left side, 2) right side, 3) front, 4) back, 5) under, 6) not close." We can paraphrase the question once the user has learned the numerical answer (by answering it several times), for instance saying just, "Car parked", to which the user can immediately respond with a number. This is a substitution code like SPIN, but it has an advantage that it can be naturally learned.

There are undoubtedly other methods that can be proposed. All will have security and usability issues and tradeoffs. Since it is likely that no single method will be best for all spoken password applications, several methods may be practical.

Acknowledgments. The authors wish to thank Colin Mallows for his invaluable help in working the probabilities of the methods and his insights into their comparative security merits, and Sachin Garg and Navjot Singh for their security analysis and suggestions.

References

1. Haller, N.: The S/KEY One-Time Password System. In: Proc. ISOC Symp. Network and Distributed System Security, San Diego, CA (February 1994)
2. Haller, N., Metz, C., Nesser, P., Straw, M.: A one-time password system. Internet RFC 2289 (1998)
3. Weiss, K.P.: Method and apparatus for positively identifying an individual. U.S. Patent 4720860, January 19 (1988)
4. O'Gorman, L., Bagga, A., Bentley, J.: Call center customer verification by query-directed passwords. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 54–67. Springer, Heidelberg (2004)
5. O'Gorman, L., Bagga, A., Bentley, J.: Query-directed passwords. Computers and Security 24(7), 546–560 (2005)
6. Ellison, C., Hall, C., Milbert, R., Schneier, B.: Protecting secret keys with personal entropy. J. of Future Generation Computer Systems 16(4), 311–318 (2000)
7. Frykholm, N., Juels, A.: Error-tolerant password recovery. In: Samarati, P. (ed.) Eighth ACM Conference on Computer and Communications Security, pp. 1–8. ACM Press, New York (2001)
8. Just, M.: Designing and evaluating challenge-question systems. IEEE Security and Privacy 2(5) (September/October 2004)

9. Dhamija, P., Dhamija, R., Perrig, A.: Déjà Vu: A user study using images for authentication. In: 9th USENIX Security Symposium (2000)
10. Kahn, D.: The Codebreakers, The Story of Secret Writing, Scribner, NY (1996)
11. Bond, M., Danezis, G.: The dining Freemasons (security protocols for secret societies). In: 13th Int. Workshop on Security Protocols, Cambridge, England, April 20-22 (2005)