Fourth-Factor Authentication: Somebody You Know

John Brainard
RSA Laboratories
Bedford, MA, USA
jbrainard@rsasecurity.com

Ari Juels
RSA Laboratories
Bedford, MA, USA
ajuels@rsasecurity.com

Ronald L. Rivest
MIT CSAIL
Cambridge, MA, USA
rivest@mit.edu

Michael Szydlo
RSA Laboratories
Bedford, MA, USA
mszydlo@rsasecurity.com

Moti Yung
RSA Laboratories
Bedford, MA, USA
myung@rsasecurity.com

ABSTRACT

User authentication in computing systems traditionally depends on three factors: something you have (e.g., a hardware token), something you are (e.g., a fingerprint), and something you know (e.g., a password). In this paper, we explore a fourth factor, the social network of the user, that is, *somebody you know.*

Human authentication through mutual acquaintance is an age-old practice. In the arena of computer security, it plays roles in privilege delegation, peer-level certification, help-desk assistance, and reputation networks. As a direct means of logical authentication, though, the reliance of human being on another has little supporting scientific literature or practice.

In this paper, we explore the notion of *vouching*, that is, peer-level, human-intermediated authentication for access control. We explore its use in emergency authentication, when primary authenticators like passwords or hardware tokens become unavailable. We describe a practical, prototype vouching system based on SecurID, a popular hardware authentication token. We address traditional, cryptographic security requirements, but also consider questions of social engineering and user behavior.

Categories and Subject Descriptors

H.m [**Information Systems**]: [Miscellaneous]

General Terms

Security, Human Factors

Keywords

authentication, hardware tokens, vouchers

CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.

1. INTRODUCTION

Passwords remain the most common mechanism for user authentication in computer-security systems. Their various drawbacks, like poor selection by users and vulnerability to capture, are prompting a rapidly mounting adoption of hardware authentication tokens. Despite stronger security guarantees, though, hardware tokens share a limitation with passwords: inconsistent availability. Users frequently forget passwords. Similarly, they often lose, forget, and break their hardware tokens.

As a result, a workable authentication system requires at least *two* modes of authentication. There is the primary mode of authentication, the password or token employed by the user in the normal course of events. Then there is the form of *emergency* authentication for cases when the primary authenticator is unavailable to a user.

On the Internet, the most common form of emergency authenticator is e-mail. When a user forgets her password for a particular site, she often has the option of having the password itself or password-reset instructions sent to a pre-registered e-mail account. Another common emergency authentication mechanism is "life questions." The user is prompted to authenticate herself by furnishing answers to previously registered personal questions, e.g., "What is the name of your first pet?" In corporate environments, the preferred emergency authentication mechanism is the help desk: employees telephone support help-desk personnel for assistance in re-establishing their access privileges. Some consumer Web sites offer this option as well.

An authentication system is, of course, only as secure as its weakest component. It is desirable, therefore, that an emergency authenticator provide security as least as strong as a good primary authenticator. Life questions, as we explain later, often fall short in this regard, as their answers are vulnerable to guessing by attackers and sometimes subject to attacks involving mining of public databases. Similarly, since e-mail is rarely encrypted, and e-mail accounts are often password-protected, e-mail is generally an inadequate emergency authentication mechanism in systems where the primary authenticator is a hardware token. Help desks can provide emergency authentication with any of a range of security assurances, depending on the manner in which staff identify callers. Help-desk staff can ask life questions, identify the voices of callers they know personally, verify caller-

ID, and so forth. But corporations commonly dislike help desks as a emergency-authentication mechanism because of high labor costs. Additionally, human intermediation at help desks introduces a vulnerability to social-engineering attacks.

Passwords and life questions are often categorized in the abstract as "something you know," while hardware tokens are "something you have." A third category of authenticator is "something you are," that is, a biometric. Systems that authenticate users based on physical characteristics—particularly voice and fingerprints—are enjoying ever-rising popularity. The general consensus of the security community, however, is that biometrics are not suitable as primary authenticators. Biometrics are often not secret. People publicly expose their voices and fingers in various ways on a regular basis, creating the possibility of biometric spoofing. (Various countermeasures can alleviate but probably not eliminate this problem.) Users generally don't forget or lose their biometrics permanently—but chapped fingers and laryngitis can lead to temporary loss. Finally, some biometrics, like fingerprints, require special-purpose reader hardware. Thus, despite their attractions, biometrics do not provide a comprehensive answer to the problem of emergency authentication.

1.1 Our work

In this paper, we explore a fourth category of authenticator: "some*body* you know." The use of human relationships for authentication is by no means new. In social interactions, introducing one person to another is the most common way of identifying (and implicitly authenticating) acquaintances. This is true in the physical world and also to a large extent in cyberspace: e-mail is a popular informal channel of authentication. In the realm of computer security, however, the natural mechanism of authentication through social networks has seen little in the way of direct formal use or exploration.

Our particular focus here is a process that we call *vouching.* Vouching is peer-level authentication in which one user, the *helper*, leverages her primary authenticator in order to assist a second user, the *asker*, to perform emergency authentication. To lend clarity to our discussion, we sometimes refer to the asker as Alice and the helper as Harry. Based on simple security principles and social-engineering and usability considerations, we design and describe a prototype vouching system for SecurID, a popular hardware authentication token typically used in conjunction with a user-specific PIN. In our system, a helper can use her SecurID token to help grant temporary access privileges to an asker who has lost the ability to use her own SecurID token, but who remembers her PIN. In a nutshell, the helper obtains a temporary passcode that we call a *vouchcode.* The helper furnishes this vouchcode to the asker as a substitute for the asker's SecurID token, i.e., as a kind of replacement one-time passcode. Using the vouchcode, the asker can authenticate without her token.

While vouching is straightforward at a loose conceptual level, a number of subtle design issues arise on closer scrutiny that do not apply to traditional authenticators. A (randomly assigned) password, for instance, carries a certain measurable level of entropy; some rough characterization is possible even when a password is user-selected. Similarly, the bit-length of the cryptographic key in a hardware authentication token, the underlying cryptographic primitives, and the output format permit rigorous characterization of the token's security properties. Vouching systems rely on traditional primary authenticators like PINs and passcodes—but in a more complicated context.

Usability and social-engineering considerations are particularly tricky facets of voucher-system design. A voucher system that is difficult to use will merely drive users to call help desks; a system that is too easy to use could evolve into a primary authenticator, eroding user vigilance around the vouching process. It is also important to ensure that askers properly authenticate the people that they are vouching for. If helpers respond to vouching requests via e-mail or to vouching requests from strangers over the telephone, then a vouching system will offer little real security. For such reasons, a vouching system must rely on trustworthy interactions within a tight social network. The problem of creating a vouching system that places helpers in a strong position to attest to the identities of askers is an important aspect of our work.

1.2 Organization

In section 2, we survey the literature on concepts related to our vouching proposal. We treat the problem of modeling in section 3. We propose and analyze a vouching system for hardware tokens in 4 and describe a prototype implementation for SecurID tokens in section 5. In section 6, we discuss social engineering and related issues. We conclude in section 7 with thoughts on further avenues of research.

2. RELATED WORK

2.1 Backup authenticators

The limitations of passwords as authenticators are well known in the security community—from the general problem of poor selection by users [12] to the disquieting inclination of users to reveal their passwords to strangers [13]. Phishing, the fraudulent use of e-mail to capture user passwords (and other information), has exacerbated the problem of password capture. Researchers have proposed a variety of countermeasures to these problems, such as browser extensions that hash passwords with domain names [17].

Life questions are another "something-you-know" authenticator, but one that serves typically in emergency authentication. Life questions have received considerably less study than passwords. Some of the security vulnerabilities of password systems, like pharming, are also applicable to life questions. Life questions, though, have their own particular features. The fact that they serve most often as a emergency authenticator—and thus receive infrequent use—can render them difficult for users to answer consistently. Moreover, as shown recently by Griffith and Jakobsson, the answers to certain popular life questions are vulnerable to attacks that involve mining of public databases [9]. The answers to many commonly used life questions, like "What was the make of your first car?" have little underlying entropy.[1] Finally, users show a surprising willingness to divulge personal information to strangers in ways that can undermine the security of life-question systems. In a recent "live phishing" experiment, in which passersby in New York City's Central

[1] General Motors, for example, had about a 43% market share in the United States in 1983 [11].

Park were offered T-shirts in exchange for filling out surveys, more than 70% divulged their mother's maiden name, while more than 90% revealed their place and date of birth [6].

Consumers' mobile phones offer a platform for emergency authentication that is increasingly favored by financial institutions for high-risk online transactions (but less often for password recovery). Mobile-phone based authentication can operate in several ways. A financial institution can initiate a call to the phone of a customer to request transaction confirmation via an automated voice recognition or keypad-based entry system—or even by means of (biometric) speaker recognition. Alternatively an institution can transmit an authentication code to a phone via SMS messaging, and request that the user enter the code into a web form [16]. These techniques are excellent adjuncts to more traditional forms of authentication, and caller ID is a simple and attractive form of authentication (although somewhat vulnerable to spoofing). They can also be useful in emergency authentication, although such use is limited at present—particularly within corporations, where mobile phones are generally not under administrator control.

Password-reset via help-desk calls is a very common practice, but it's expensive. Vendors of password-reset products claim that each password-reset request involving live staff costs some $15-30 [1, 3]. This cost alone is a strong impetus for creating alternatives. Moreover, because help-desk calls often involve interactions between strangers, they are vulnerable to social-engineering attacks, such as those described by Kevin Mitnick in his *Art of Deception* [15].

2.2 Social relationships and authentication

Reliance on "somebody you know" is an age-old vehicle for authentication in everyday life. When you introduce one friend to another, you are effectively performing an authentication protocol based on a social relationship. Social-network-based authentication also has a pervasive but largely informal role in the security infrastructures of organizations. When an employee holds open an access-controlled door for a familiar colleague, when a system administrator resets a password for a colleague whose voice she recognizes on the phone, or when a manager brings a new employee to a corporate badging center, a form of vouching is taking place.

Familiar social interactions of this kind have given rise to analogous systems in cyberspace. People regularly send e-mail to colleagues and friends in order to effect introductions and request grants of privileges (to set up accounts, make payments, etc.). *Reputation systems* are an extension of the vouching principle to large on-line communities; eBay provides a familiar example in which account holders provide mutual ratings of commercial integrity. Given their loose authentication of participants, though, reputation systems are unsuitable mechanisms for authentication of individual identities.

Peer-level public-key infrastructures (PKIs) such as PGP and SPKI/SDSI [2, 8] provide a more rigorous basis for authentication through social networks. In such systems, principals make local decisions about whether to endorse the identities (certificates) of other principals. In the case of PGP, a social network called a "Web of Trust" helps authenticate e-mail addresses. Human intermediation can serve as a component in peer-level (and hierarchical) PKIs. It is a straightforward matter to define a credential asserting attestation of identity based on human contact ("I have met so-and-so face-to-face") and to create supporting policies and software interfaces. Trust-management systems like PolicyMaker [5] and its successor, KeyNote [4], permit general policy decisions around access to resources on the basis of digitally signed credentials, and can in principle take human relationships into account.

PKIs and trust-management systems, however, are abstractions for the creation and management of digital credentials. In contrast, in this paper we consider "somebody you know" as a starting point for authentication; our investigation is predicated on fundamental usability and human-interaction issues, rather than reference to a particular cryptographic mechanism. In this view, Carl Ellison's concept of "ceremonies" is particularly important prior work [7]; Ellison proposes a model to capture the human behavior surrounding cryptographic authentication protocols. Also pertinent are recent authentication systems designed with human factors as first principles. A recent example is "Seeing is Believing," a system that exploits visual contact (2D barcodes) as a physical mechanism for trust [14].

3. MODELING

In this section we model the security properties that we would like a voucher system to achieve. We begin by describing the parties and communication channels involved in the system. Then we discuss our assumptions concerning the two pieces of authentication data used. These items are the PIN and the tokencode. We describe the types of adversaries we intend to protect against, and present security requirements for such adversaries.

Parties and Channels: The principal parties involved in the primary authentication mechanism are the *User* and the *Server*. For the vouching protocol two types of users play distinct roles. These are the *Asker* and the *Helper*. We also consider a malicious external party *Adversary* with various capabilities.

Users or parties are denoted by a capital letter (e.g., X), and to each party we associate an identifier, which will be denoted with the corresponding lowercase letter. For example, the identifier of a user X will be denoted by x. Within an invocation of a protocol among users, we use the notation $X(y)$ to indicate that party X has represented itself as having claimed identifier y. With this notation, an honest party X will always be represented as $X(x)$. Our notation suggests that parties use consistent identifiers throughout the protocol, and we make a remark whenever this may not be the case.

Users have certain initial conditions imposed on them that can be expressed as relations. Namely, some users are enrolled as helpers, and users are organized in a (helper, asker) relation $\mathbf{H}$ where each helper is assigned a subset of users as askers; we say $(Y, X) \in \mathbf{H}$ if a party X is an asker for party Y acting as a helper.

In our model, the communication channels to and from the server are considered to be secure. In particular, this means that no adversary can obtain any information by eavesdropping over such channels. In practice, channels between an entity and the server may be achieved with SSL. The channel between an asker and a helper may be less secure, and we need to carefully design our protocol setting to mitigate eavesdropping risks on this channel.

Authentication Data Items: The basic authentication protocol as well as the vouching protocol involve two primary factors that the user presents to the system in order to be accepted as authenticated. The basic two factors are the PIN, PN, which is a user memorizable string, and the tokencode, TK, which is generated from the hardware token. Other factors are possible. In particular, a third piece of authentication data is the vouchcode, an ephemeral item used within the vouching protocol.

To model the information made available in a protocol message by parties, we introduce the notation $(X(u) : VA)$ to indicate the event wherein a party X sends a message claiming identity u and correctly presents the value item of VA. The following assumptions concern the probabilities of an adversary being able to present the user's PN and TK at any point within the protocol.

1. For any party X and any identifier y, the probabilities $prob\{X(y) : PN\}$ and $prob\{X(y) : TK\}$ are independent.

2. The legitimate user U who chose the PIN value and possesses the token and enrolled in the system as u can always present the PIN and tokencode. That is, $prob\{U(u) : PN\} = prob\{U(u) : TK\} = 1$. These assumptions amount to a user being able to recall their PIN value and also being in posession of their token.

3. An adversary, A, who is not U, and has not obtained U's PIN value PN, can only present it with small probability. That is $prob\{A(u) : PN\} < \epsilon$.

4. An adversary, A, (where $A \neq U$), can only present a tokencode with small probability. That is $prob\{A(u) : PN\} < \delta$.

Note that the adversary's probability of successfully presenting the required values (PN or TK) is the probability bound on the union of two events: (1) the adversary guessing the value (which is typically assumed long enough and to contain certain amount of entropy), and (2) the adversary stealing or otherwise obtaining the value.

Authentication Ceremony: An authentication ceremony AC is a sequence of interactions between a number of parties. It was introduced to describe not just a protocol between the user software agent and the system, but to designate that a party may act via his software agent and also act personally. Once defined, a ceremony is invoked by various parties (with certain relationships among them) who follow a sequence of actions. We use notation AC_T to refer to a ceremony of type T or an instance of this type of ceremony. To specify the parties P_i and the identifiers p'_i presented by these parties we use the notation $AC_T(P_1(p'_1), P_2(p'_2), \ldots)$.

In an authentication ceremony parties exchange messages via secure channels, and one of the parties will be the server S. The server is assumed to be trusted and its role is to accept the legal executions and reject the adversarial ones. The correctness and security definitions of an authentication ceremony are defined in terms of the server's final state, which is always either "accept" or "reject."

Interactive Logging and Detection: Given a ceremony, we may consider an extended ceremony where in addition to message channels there exist also "logging channels" where parties post information "about" the ceremony: e.g., posting the time of the invocation, the parties, and the state of the execution). These logging channels are not erasable by the parties, and can be viewed as a bulletin board maintained by the trusted server, and can be implemented by getting e-mail upon demand from the server.

By considering the extended ceremony that includes the original messages that are designed to stop impersonation (i.e., to reduce the probability of an attack), logging and log evaluation further allow for the detection of (what we hope are rare) successful attacks. For example, a party evaluating the log may detect unexpected behavior such an authentication under its identifier in which it did not take part. We denote the logging channels available to party U as $LG(U)$. The party reads and evaluates the log and decides whether there as been an attack or not. If an attack is detected in this way appropriate countermeasures can be taken. In actual implementation, the logging functionality may be implemented as a combination of e-mail notifications and server logs.

Next we discuss correctness and security definitions.

3.0.1 Correctness

The correctness requirement of the system is that when the parties act honestly, the authentication attempt will succeed, i.e., for a given authentication ceremony $AC = AC(P_i(p_i))$ (where in all cases party P_i and its true identifier p_i are involved and where the parties satisfy the initial required relationships) then the server "accepts."

3.0.2 Security Properties

To describe the security properties of a ceremony, we define adversaries (outsiders and insiders) who attack other parties. In an authentication ceremony, the adversarial goal is to impersonate a party. There are two types of security properties: (1) Prevention: where the server accepts an invocation with a party claiming a wrong identifier only with small probability. (2) Detection: where successful attacks are realized by parties.

4. VOUCHER SYSTEM

In this section we describe a voucher system for hardware authentication tokens such as RSA Security Inc.'s SecurID. A SecurID token typically takes the form of a key fob or card that displays a fresh numerical value, called a *tokencode* or *passcode*, every sixty seconds. To authenticate to a computer application, the user must type the current tokencode together with a user-specific PIN or password.

The tokencodes in a SecurID installation are validated by a specially designated authentication server. This server shares a secret seed with every token and also the PIN of the user, allowing it to validate tokencode/PIN pairs. The technical details of tokencode computation are largely unpublished. Briefly, though, a tokencode is computed as a cryptographic function of the current time and a secret key shared between the token and authentication server [10]. Our system can be applied equally well to other token-types, e.g., Verisign authentication tokens.

We will describe the regular system AC_R and the vouching system AC_V. In both cases we will describe the messaging and the extended ceremony with logging.

4.1 Regular Authentication

We first review the regular two-factor authentication protocol, AC_R, which does not include vouching. This illustrates the basic system upon which a vouching system will be constructed.

Enrollment: The process gives each user a PIN value and a hardware token that at each time unit produces tokencodes. At the server side the user identifier is associated with its chosen PIN and with the specific tokencode. Thus the server is able to produce the authentication values and compare them to the ones presented by the user.

Authentication Session: Once enrolled, a user U authenticates herself by presenting her identifier u, her PIN PN and the current TK to the server via a secure channel. Once the server gets the values it checks them against computed/stored values. If the values match the server "accepts" and otherwise it "rejects."

If logging is implemented, the server logs to $LG(U)$ that a session with u has taken place at its given time unit. This logging event is available to user U (and is not manipulated by it), and U can evaluate the log.

Properties of the regular protocol:

- Correctness: for the legitimate user U claiming to be itself $prob\{U(u) : PN\} = prob\{U(u) : TK\} = 1$; thus the server will always accept.
- Security: (prevention) The adversary A performs the authentication session with identity u which is not his own. Since $prob\{A(u) : PN\} < \epsilon$ and independently, $prob\{A(u) : PN\} < \delta$, the server accepts with probability at most $\epsilon\delta$, which is very small, thus impersonation is prevented.
- Security: (detection) In case the server maintain logs that are accessible by the user at all times, in the unlikely event that A is successful, U will learn about a session initiated under her identity at a time when she herself did not participate. Thus she will detect the break.

4.2 Concrete protocol steps

The vouching system we describe is designed to deal with the case in which a user does not have her token available, but does recall her PIN. In other words, it is designed to deal with unavailability of one of the two authentication factors. Expressed succinctly, the vouching process involves this user (asker) contacting a pre-registered helper for assistance in authenticating herself. The helper obtains a temporary tokencode from the server, called a *vouchcode*. The helper communicates the vouchcode to the asker. The vouchcode aids the asker in the emergency authentication process. The vouching system involves the server and the user (or asker) and helper. The vouching ceremony is denoted $AC_V = AC(S, U, H)$. We now explain in detail the steps involved in the vouching process.

Enrollment: In this stage formally the relation $\mathbf{H}(X, Y)$ is created. The server records this relation and each user learns its helpers and its askers.

It is a matter of system policy which users may act as helpers for a given asker—and how those helpers are designated. One possibility is for an asker to be enrolled in the voucher system concurrently with the original provisioning of the hardware token. At this time an appropriate administrator (or perhaps the asker's supervisor) is notified by e-mail to take action. This notification directs the administrator to a management interface through which the administrator can explicitly specify which helpers are be allowed to vouch for this asker. The same management interface allows for subsequent modifications to the association of helpers to askers. Each helper is automatically sent e-mail notifying him of this responsibility, and must acknowledge and agree to the corporate policy concerning vouching. This may be implemented in a very convenient way, for example embedding an "accept" button in the e-mail notification. We believe that the explicit nature of assuming responsibility as a helper is central to clarifying the accountability of security risks.

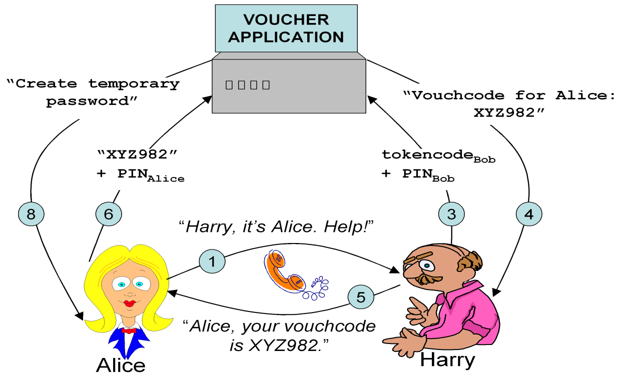
VOUCHER APPLICATION
"Create temporary password"
"XYZ982" + PIN$_{Alice}$
tokencode$_{Bob}$ + PIN$_{Bob}$
"Vouchcode for Alice: XYZ982"
8
6
3
4
"Harry, it's Alice. Help!"
1
5
"Alice, your vouchcode is XYZ982."
Alice
Harry

Figure 1: A schematic of the basic vouching process: Harry the helper aids Alice to obtain a temporary password. Step numbers correspond to those in text description (and some are omitted).

The process of assigning helpers to askers can also be partially automated by populating the management interface with existing relationship data, such as that contained in corporate organizational charts. A variant of the enrollment procedure might allow askers to choose their own helpers, although this could increase risks if a supervisor is not involved.

Once enrolled, an asker who has lost the ability to use her token may cooperate with a helper to achieve emergency authentication via the following steps:

1. Asker Contacts Helper: The vouching process begins with the asker U contacting the helper H via an out-of-band channel and claiming to be u. The channel can be the telephone or even face-to-face contact. As we explain later, e-mail contact should be deprecated or prohibited.

2. Helper Authenticates Asker: The helper verifies the identity of the asker. Formally, the process has to assure H with probability at least $1 - \mu$ (for small μ) that the claimed identifier u is the asker's true identifier. In implementing this step over a telephone communication, the helper makes sure that he recognizes the asker's voice (and her telephone, if possible, using caller-ID); the helper may, if in doubt, ask questions that help confirm the asker's identity. In the case of face-to-face contact, the helper authenticates the asker as a matter of course. It is also assumed that the asker identifies the helper and that they both recognize that they are in a helper-asker relationship.

3. Helper Authenticates to Server: Using his own client machine, the helper accesses a vouching-specific web page. Through this interface, the helper strongly authenticates himself to the server using two factor authentication—

i.e., using his token TK and PIN value PN as in the regular authentication session described above. The helper asserts identity h and declares that he is helping user u.

4. Helper Obtains Vouchcode: In response to successful authentication, the helper is prompted by the server S for the name of the asker. The server verifies that H and U are in the helper-asker relationship. If the verification succeeds, the helper then receives an asker-specific *vouchcode* VC (we assume that VC is guessable with probability at most δ) . The server further marks the fact that there is an ongoing vouching session involving H and U. The vouchcode assumes an alphanumeric form amenable to verbal communication. For easy communication, it should also be relatively short; it contains 20 bits of entropy in our prototype.

5. Helper Gives Vouchcode to Asker: Next, H informs U of the value VC —orally in the case of telephone communication, and either orally or as a written value in the case of face-to-face contact.

6. Asker Enters Vouchcode: U asserts identity u and presents his PN and VC to S. In our implementation, using a special-purpose web interface on her own client machine, the asker authenticates by entering her username, the vouchcode, and her PIN.

7. Server Authenticates Asker: Upon receipt of this information, the server identifies in its database an active vouching session for this asker. A vouching session is considered active if it has been initiated by a valid helper within a short time period as specified by the system policy (e.g., 3 minutes). (This timeout helps ensure tight synchronization of the vouching process, and that vouchcodes are not saved for later use.) The server verifies the asker's PN and VC. The server either "accepts" the authentication attempt or "rejects", i.e., aborts the vouching session; either way, the asker is informed of the result.

8. Temporary Password: If the asker has successfully authenticated, she is prompted to choose a temporary password TP and granted access for the current session. The asker can use this temporary password together with PN for subsequent sessions, while the vouchcode becomes invalided. The asker's temporary password expires after a time period specified by system policy (e.g., one or two days).

One important reason to have the user convert her vouchcode into a temporary password is that a user-selected password is likely to be easier to remember than a system-generated vouchcode. We touch on security-related reasons in our discussion below.

9. Logging: An important component of a voucher system is its support for detection mechanisms based on user notification and administrator audit. We propose that confirmations of successful/failed asker authentication be sent to the helper and to the asker (and, as a system option, to an administrator). Additionally, an audit log should record all transactions in the vouching system.

Briefly, then, in order to obtain a vouchcode for use by the asker, the helper makes use of the registered helper-asker association and his ability to perform strong authentication. The asker employs this vouchcode and her PIN to obtain a temporary password. This password serves effectively as a token substitute: the asker may use it in lieu of a tokencode wherever system policy permits.

4.3 Defining and Claiming the Voucher System Properties

4.3.1 Correctness

The **correctness** property requires that a legitimate user U (claiming to be u) when asking H (claiming to be h) for help will will always be successful in getting the server S to accept provided $\mathbf{H}(H, U)$. To see that correctness holds, we simply examine the steps of the protocol.

1. In steps 1 and 2, H and U will recognize each other and U has a way to assure H of her identity with probability 1.

2. In steps 3 and 4, H will cause S to accept its authentication with probability 1. This is based on the correctness claim of the regular authentication protocol.

3. Successful authentication will result in a vouchcode VC that will get to U in step 5. The server S is expecting this value within a vouching session. Since $\mathbf{H}(H, U)$, S will produce a VC associated with H and U with probability 1.

4. In steps 6, 7, and 8, U will be able to present her identifier u, PN, and VC correctly to the server, which will accept with probability 1. Furthermore, U will be able to enter his chosen temporary password TP.

5. Finally, the log messages will be always be received by U and H, who expect them as a positive feedback.

4.3.2 Security

We now discuss the **security** of our proposed vouching system by enumerating several formal properties and discussing specific types of attacks. Practical security concerns such as social engineering will be treated in the next section.

We assume in our discussion here that an adversary does not benefit from collusive attacks—as should be the case in well-designed systems enforcing independence among user PINs and authenticators. In fact, when an adversary is attacking a user in the system, we may assume it controls all other users who are not involved in the session and their secrets and we merely concentrate on a specific invocation. We also assume the trustworthiness of the enrollment process, and the system administration and server S. If these assumptions did not hold, the primary authentication mechanism itself would be vulnerable.

To describe the desired security properties we need to first clarify the capabilities of the Adversary. By an *Outside attacker* we mean an attacker who is not enrolled in the systems as either a user, or a helper. By an *Inside attacker* we mean an attacker who is either an asker, helper, or another inside party who acts under a false identifier or without the preconditioned relationship holding.

Prevention Requirements: In an authentication ceremony, the most important aspect of security is to prevent unauthorized access. We define the prevention aspect of security in terms of the following adversarial invocations. Any adversary taking part in a protocol in any of these instances should lead to S rejecting with high probability.

- (User impersonation by an outsider): An instance where $AC_V(S, A(u), H(x))$ where $A \neq U$ should not be accepted. The actual helper H may or may not be present.

- (Helper impersonation by an outsider): An instance where $AC_V(S, U(x), A(h))$, where $A \neq H$, should not be accepted. Such an instance should fail regardless of the user.

- (Helper not registered:) When $(U', H') \notin \mathbf{H}$, an instance of the form $AC_V(S, U'(u'), H'(h'))$, should not be accepted. Such an instance should fail even if U' and H' are colluding.

- (H attacking U:) An instance $AC_V(S, H(u), H(h))$, where H is an adversary trying to impersonate U, should not be accepted.

- (U attacking H:) An instance $AC_V(S, U(u), U(h))$, where U is an adversary trying to impersonate H, should not be accepted.

Detection Requirements: The following detection properties deal with the case in which an attacker has managed to get the server to accept the authentication attempt. In such a case, detection is a second line of defense.

- (Detecting user impersonation by an outsider): In case of a successful instance where $AC_V(S, A(u), H(x))$ with $A \neq U$ (user impersonation, with or without the actual helper H being present), U will detect it in the log.

- (Detecting helper impersonation by an outsider): In case of a successful instance $AC_V(S, U(x), A(h))$, where $A \neq H$, i.e., helper impersonation by a third party, regardless of the user, H will detect it in the log.

- (Illegal help request:) In case of a successful instance where $AC_V(S, U'(u'), H'(h'))$ where (U', H') not in $\mathbf{H}$ (U' and H' are colluding in this attack), U and H will detect it in the log.

- (Detecting H attacking U:) In case of a successful instance where $AC_V(S, H(u), H(h))$, where H is an adversary trying to impersonate U, U will detect it in the log.

- (Detecting U attacking H:) In case of a successful instance where $AC_V(S, U(u), U(h))$, where U is an adversary trying to impersonate H, H will detect it in the log.

Next we discuss how the above definition is satisfied by the protocol.

Outside Attacker Impersonates User: We first consider the case of an outside attacker who targets user U enrolled in the voucher system as an asker. Of course, the attacker can—irrespective of the presence of the vouching system—impersonate U by obtaining or guessing her current TK and PN and using the regular authentication session, but this was shown to be an event of small ($\epsilon\delta$) probability.

The adversary A may start an instance as user U, with $A \neq U$, but he will fail with overwhelming probability to convince H that he is U in step 2.

The adversary might also try to obtain U's vouchcode and PIN. The first way for the attacker to obtain U's vouchcode is by guessing it (knowing that U has asked for help and a VC exists). But guessing PN and VC is not easier than guessing TK and PN.

In the unlikely event that the user has been impersonated, she will be able to detect it in the log.

Outside Attacker Impersonates Helper: Next the attacker may attempt to impersonate the helper. This is an instance of the protocol $AC(S, U(u), A(h))$ where $A \neq H$. In this case step 3 will fail with probability $1 - \epsilon\delta$. Again, in the unlikely event of success the helper will detect the attempt in the log.

Unregistered Helper: Another attack on the system can be an instance where the user U' and the helper H' do not have the precondition relation $(H', U') \in \mathbf{H}$. While a helper can obtain a vouchcode for a user who is a registered asker for H', the system will not produce a vouchcode in step 4 for user U' who is unregistered. In the unlikely event that a legal helper H is impersonated by an adversary, H will recognize an interactive logging event that he has not actually taken part in.

Helper attacks Asker: Once an asker is assigned a helper in the enrollment process there is a risk that the helper attacks the asker. H can obtain a vouchcode for U without even involving her. By guessing U's PN (with probability ϵ in our model), H can then impersonate the asker in the voucher system.

Briefly, then, the main new risk that the vouching protocol introduces is that security against malicious helpers is reduced to that of obtaining the asker's PIN. In case of a successful attack, U finds a session in the log in which she did not participate.

We remark that in our definition of an authentication ceremony we assume that a party presents itself with a consistent identity throughout the protocol instance. In reality, in the vouching ceremony, an adversary acting as a helper may present itself to a user under one identity and to the system under another identity. The analysis of such attacks with divergent identities in very similar to our analyses with consistent identities.

Asker attacks Helper: When an asker U attempts to attack the helper in our model, it can only ask for help, authenticate itself, and obtain a vouchcode. No helper-specific information is leaked or made available to the asker. Thus, an asker can only attack a helper as an outsider, and no better.

4.4 Pragmatic consideration of Attacks Beyond the Model

Remarks on Outsider Attacks: We have formally modelled attacks on vouchcodes via a guessing or misbehaving adversarial helper. We note that in practice an avenue for an outside attacker is to obtain a vouchcode in transit, for example if it is communicated over insecure e-mail from H to U. This is one reason why our system discourages or prohibits e-mail on the asker-helper channel. As we discuss below, an attacker can alternatively attempt to extract U's vouchcode from H via social engineering. To ensure that such refined attacks do not weaken the vouchcode system, it is desirable that their collective probability be at most a small value δ. (Quantifying this probability rigorously in real life systems is a challenge, of course.) One reason to have the asker convert a vouchcode into a passcode (i.e., step 8 of our vouching protocol) is to minimize the risk of vouchcode compromise by an outside attacker. If, for example, the helper provides the vouchcode to the asker on a slip of paper, then it is particularly desirable that the vouchcode expire quickly.

Remarks on Illegal Helping: The system should avoid situations where it is easy for a helper to get the PIN of the asker, e.g., when the asker is tempted to use the helper machine (for example in the helper's office) in order to get a temporary password, while the helper has a malicious software that logs and snatches the asker's PIN.

Remarks on Asker Attacking Helper: Playing the role of the helper in an actual real world setting carries some risk. U could ask H to vouch for her—i.e., request a vouchcode on her behalf—on her own machine. U can make a compelling case for this, since it may be convenient or make H look impolite if he refuses. U, however, if malicious, could have keystroke-logging software on her computer that captures H's PIN.

Such attacks are even more of a concern when two parties are enrolled to vouch for one another. As discussed above, it is enough for a helper to know a user's PIN in order to impersonate her.

Of course, if H authenticates on U's machine—or any other untrusted machine—for any purpose, he exposes himself to the possibility that hidden malicious software will capture his tokencode and PIN in real time and impersonate him completely. This problem is not specific to vouching: an organization that allows any sharing of machines exposes itself to such vulnerabilities whether or not a vouching system is in place. It is possible, though, that the presence of a vouching system may encourage sharing of machines. Some countermeasures to asker-helper attacks are possible. For example, the vouching system can enforce a policy whereby an asker is removed from the helper's list of helpers until the helper changes his PIN.

5. AN IMPLEMENTATION

RSA Laboratories developed a prototype implementation of Voucher-based authentication. The system consisted of three components: an asker web application, a helper web application, and an administrative program. Both web applications consisted of sequences of Common Gateway Interface (CGI) applications developed in C++ and running on the Microsoft® Internet Information Services (IIS) web server. The administrative application was a Windows® application developed in C++ using the Microsoft Foundation Classes (MFC). The underlying database, common to all the applications, was built in Microsoft Access.

The asker application simulates a generic web application protected by RSA SecurID®. The subsequent pages walk the asker through the steps required to authenticate without possession of a token. These web pages are as follows:

- Asker Page 1: The normal user login page for SecurID, with fields for user name and PASSCODE. The special, added feature for vouching is a button labeled "Forgot/lost my Token."

- Asker Page 2: Instructions for asker to contact a helper, with a form to enter the helper's login name.

- Asker Page 3: A form for entering the asker's PIN and the vouchcode supplied to the asker by the helper.

- Asker Page 4: If the PIN and vouchcode are both determined to be correct, this page displays another form that allows the asker to enter and confirm a temporary password. (The system may enforce security restrictions on the form of the password. For example, in our prototype, passwords must be at least eight characters in length.)

- Asker Page 5: If the password is confirmed, the asker is presented with another login page where the PIN and temporary password are used to log in.

- Asker Page 6: If the PIN and temporary password are both verified successfully, the asker is presented with a page confirming successful authentication.

The helper application allows the helper to retrieve vouchcodes for askers. It consists of only two pages:

- Helper Page 1: The helper is presented with a login form with fields for his user name and passcode, along with the name of the asker being assisted.

- Helper Page 2: If the helper's passcode is successfully verified, a vouchcode is generated for the specified asker. The vouchcode is displayed to the helper who is then responsible for conveying the value to the asker.

The administrative application controls overall system configuration and the vouching capabilities of individual askers. The application uses two dialogs, a general "Voucher Management" dialog and an asker-specific "Helper Management" dialog. The Voucher Management dialog controls several system parameters, such as the length and validity periods of both vouchcodes and temporary passwords. The Helper Management dialog specifies rules for the set of permissible asker-helper relationships. In our prototype, these relationships are confined within groups that are pre-defined by an administrator database. In a real application, this database might be imported from an external source, such as an HR database, in order to take advantage of previously established groups. In our prototype, a given user may be designated as permitted to help anyone in the same group, permitted to help selected askers in the same group, or prohibited from helping any askers.

5.1 Implementation issues

Two significant questions arose in the implementation of our prototype voucher system. The first is this: should the application provide the asker with a list of her helpers? The difficulty, of course, is that since the asker has not yet authenticated to the system, such a list would be accessible to anyone. Helper lists could open the system up to social engineering attacks, since an attacker could learn the list of helpers for an asker he wishes to impersonate. We could, of course, require an asker to enter her PIN in order to see her list of helpers. This approach, however, would expose the PIN to a new vector of guessing attack. Omitting helper lists entirely could prove problematic because it may not be reasonable to expect askers to remember all of their helpers, especially if vouching is used infrequently. In our prototype, we have chosen not to list helpers, but this should in general be a matter of policy.

The other question involves the seeding of suitable groups in the Helper Management application. In an organization of any size, having individual helpers authorized to vouch for all askers is problematic, since there may be many askers whom a given helper does not know. A logical alternative is to have supervisors act as helpers for their subordinates, but this may prove to be an unacceptable burden for the supervisors—and an embarrassment for their subordinates. Our conclusion is that the most suitable arrangement is for groups to consist of organizational peers. This should work well: we can generally assume the askers within a group are familiar with each other and willing to act as helpers. Some form of peer-level group structure exists in most organizations—and, indeed, most HR databases—and need not be created just for the purpose of using vouchers.

We are presently in the process of conducting a small pilot study to understand user interaction with and refine our prototype system.

6. SOCIAL ENGINEERING

The security of a voucher system depends critically, of course, on how users interact within its framework. In addition to ordinary data-security considerations, as reflected in our security model, it is vital to consider the various potential vulnerabilities to social engineering. Simple refinements to a vouching system—some of them already incorporated into our prototype—can considerably strengthen the security of the system as a whole. In this section we explore the threat of social engineering and related problems and some possible countermeasures.

6.1 Tailgating

Politeness is often at odds with security. For example, the practice of "tailgating," in which employees allow people to follow them when they unlock doors, undermines the security of physical access-control systems in office buildings. A desire for politeness often causes employees to grant access even to strangers, rather than to challenge them. A naïvely deployed voucher system can be similarly vulnerable to tailgating. If Harry receives a telephone call from a colleague Alice asking for his help, Harry may well feel uncomfortable refusing—even if he doesn't know Alice well (or at all).

In contrast to a physical-access system, a voucher system can impose restrictions on asker-helper relationships and activity. In our prototype, the administrator defines the relationships permissible in the vouching system, allowing vouching, for example, only between peers of well-defined corporate groups. The aim of such features is, of course, to ensure against situations in which Harry feels pressure to offer help to an inappropriate asker Alice.

One can imagine a broad range of alternative mechanisms for constraining helper-asker relationships. The possibilities are particularly rich when an interface exists between a vouching system and other communication systems. For instance, a vouching system integrated with a telephone system might automatically enroll Alice as an asker for Harry if the two speak on the telephone on a regular basis. (Such solutions require sensitivity to privacy concerns, of course.) Additionally, Harry might only be permitted to vouch for Alice when she calls from a telephone number that Harry himself has recently called. For deployment of voucher systems in consumer environments, where administrative oversight and user awareness of security may limited, such automated approaches could be particularly attractive. They are probably not practical in today's environments, but could become so in future, when VoIP and similar technologies result in tighter integration of communication systems.

6.2 Weakly authenticated contact

Similar in spirit to the problem of tailgating is that of weakly authenticated contact. Suppose that Alice is registered as an asker for Harry, and Harry receives a request via e-mail from someone purporting to be Alice. Should he offer his help? Such e-mail could, of course, originate with an entirely unauthenticated source, e.g., a public e-mail account. Without proper safeguards, Harry could easily end up helping an impostor.

It is important that vouching policies prohibit poorly authenticated contact by an asker. Helping should be forbidden to askers who communicate by e-mail—unless, perhaps, the e-mail is internal to a company, followed up with a phone call, or otherwise appropriately authenticated. The safest policy is to mandate telephone or face-to-face contact.

Policy enforcement, however, is a tricky matter. Ideally, as suggested in our discussion of tailgating, an integrated platform might automate policy enforcement. In existing systems, with their loose affiliation among modes of communication, such automation would be difficult to achieve. The most practical approach, therefore, is to require helpers to indicate how they have been contacted by askers, and to authorize or deny vouching transactions accordingly. We are in the process of implementing a pull-down menu (in Helper Page 1) for this purpose in our prototype; the descriptions "E-mail," "Telephone," "In person," and "Other" specify the form of asker contact. If the helper chooses "E-mail" or "Other," the system warns the helper that the mode of contact is prohibited by system policy and denies the vouching transaction.

We emphasize the importance of allowing the helper to specify the form of asker contact accurately, even if the form of contact violates system policy. Otherwise, the helper may be tempted simply to bypass the protective mechanism. For example, if a menu of contact options lists only the choices "Telephone" and "In person," users contacted by e-mail may be tempted just to select "Telephone" because of the lack of a more accurate option (in spite of any warnings). Delicate design decisions can have a big impact. Even the order of menu choices is important: the pull-down menu in our prototype gives "E-mail" as a first choice, so that careless menu selection results in rejection of the helping request.

Another mechanism that can help prevent inappropriate use of vouching requests by e-mail is reduction or elimination of the helper's ability to cut and paste voucher codes. For example, a vouchcode might be displayed as an image, rather than text. (Some people display their e-mail addresses on web pages this way to discourage bots from harvesting them.)

Of course, a user who wishes to bypass the contact policy of the vouching system can do so. Our belief, however, is that most users will willingly comply with system policy provided that such compliance is not onerous. Of course, administrator-level system auditing can be a powerful additional mechanism for policy enforcement. We believe that the e-mail notices in our prototype—a kind of peer-level audit mechanism—will also help curb abuses.

6.3 Spidering

Any systemic weakness in the second factor used alongside voucher codes (the PIN in our prototype system) poses a special hazard. Given such weakness, an attacker that has compromised one helper can readily compromise other accounts. Consider, for example, an organization in which PINs are user-selected and many users choose the PIN '1234' for convenience. An attacker who has compromised Harry's account can simulate a voucher request from Alice, perhaps guess Alice's PIN, compromise Alice's account, proceed to attack Alice's enrolled askers, and so forth. An organization with a loose social network and thus high exposure to social engineering would be similarly at risk.

The helper-asker relationships in a voucher system may be represented very simply as a digraph D in which each user is represented by a node and each enrolled helper-asker relationship as a directed edge. A natural strategy for an attacker that has compromised one node is to exploit weak adjacent edges in order to compromise as much of the system as possible. We refer to such an attack—and its exploitation of the web of helper-asker relationships—as *spidering.*

We emphasize that traversal of any edge in D by an attacker requires compromise of the second factor in the vouching system (users' PINs). Thus the best defense against spidering is a sound vouching process combined with a good second factor. Other defensive mechanisms are possible, however. Among these are:

1. *Dynamic digraphs:* The digraph D might evolve as a function not just of user relationships, but also recent voucher assignments. For example, if Alice has authenticated using a voucher assigned by Bob, Alice herself might lose her permission to assign a voucher to another employee for a certain period of time or until she re-authenticates in a stronger manner. Other throttling rules might also apply. For example, suppose that an edge in D is colored if vouching has taken place recently between the corresponding helper and asker. The vouching system might prohibit vouching or alert administrators if the diameter of a colored subgraph grows beyond some pre-specified threshold.

2. *Multiple vouchers:* To extend the digraph-based view of a voucher system, a user may be able to authenticate only as the result of a joint operation among several connected users. For example, Alice might only be able to authenticate in a voucher system on receiving vouchers from two authorized helpers, rather than one.

6.4 Lazy vouching

Of course, a voucher system should be easy to use. If we make it too easy, though, users may be tempted to rely on the voucher mechanism as a primary authenticator. We refer to such undue reliance on vouching as *lazy vouching.*

For example, Alice might decide never to bother obtaining a SecurID token from system administrators, but instead rely on Harry whenever she needs to authenticate. In some environments, this may not be a problem. After all, lazy vouching does not introduce a new technical vector of attack. Lazy vouching can undermine system security in two ways, however: (1) In a voucher system with administrative audit, lazy vouching can make vouching seem unexceptional, and thereby obscure system breaches; (2) By desensitizing users to the process of vouching and the surrounding checks, lazy vouching can increase risks of social engineering.

For these reasons, we believe that vouching systems should be designed and configured to discourage lazy vouching. In practice, we believe that the light burden of having to fill out browser forms in our prototype is a good deterrent.

7. CONCLUSIONS

This paper introduces the concept of vouching as a tool for on-line authentication. Vouching directly leverages human relationships, and this work can be seen as part of a broad exploration of the interplay between social networks and user authentication. Breaking the common conceptual identification of a user with her client machine, and common modality of single-user involvement in authentication, we have extended the classic authentication triad of "what you have," "what you know," and "what you are." In addition to introducing the concept of vouching as a tool for on-line authentication, this paper provides a rough security model for analyzing vouching, and discusses the special social-engineering concerns of a multi-user protocol. We present a specific protocol for vouching with hardware tokens, our implementation with the well known SecurID product, and thereby demonstrate both feasibility and usability.

The basic vouching system we have outlined has been designed to be as simple and secure as possible, but several variants and extensions are worth further exploration:

Alternate Factors: Vouching need not be restricted to hardware tokens and accompanying PINs. The basic vouching procedure could work with a different primary authentication factor. As an example, in the protocol we describe for SecurID, a voucher system could permit swapping of the roles of the tokencode and PIN to deal with the case where an asker has forgotten her PIN but still has her token.

Restricted privileges: To enhance the security of a vouching protocol and discourage lazy vouching, a voucher system might grant restricted privileges to askers. For example, the system might permit viewing of e-mail, but not sending, or might limit access to company-sensitive documents.

Asker Selection of Helpers: In some deployments the involvement of an administrator or supervisor in enrollment might be cumbersome. It may in some cases be desirable for askers to designate their own helpers. The security risks of this type of approach deserve further scrutiny.

8. REFERENCES

[1] v-GO SSPR 5.0 product description. Referenced 2006 at www.passlogix.com.

[2] Simple Distributed Security Infrastructure (SDSI) web page, 2001. Referenced 2006 at http://theory.lcs.mit.edu/~cis/sdsi.html.

[3] PeopleSoft and Courion deliver integrated password management solution, 27 August 2001. Press release. Referenced 2006 at www.courion.com.

[4] M. Blaze, J. Feigenbaum, and A. D. Keromytis. The KeyNote trust management system. In *Security Protocols International Workshop*, pages 59–63. Springer-Verlag, 1998. LNCS no. 1550.

[5] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance-checking in the PolicyMaker trust-management system. In *Financial Cryptography*, pages 251–265. Springer-Verlag, 1998. LNCS no. 1465.

[6] W. Eazel. 'Live phishing experiment nets consumers hook, line, and sinker. *SC Magazine*, 8 November 2005. Referenced 2006 at www.scmagazine.com.

[7] C. Ellison. UPnP security ceremonies design document: For UPnP device architecture 1.0, 3 October 2003. Referenced 2006 at http://www.upnp.org.

[8] C. Ellison. IETF RFC 2692: SPKI requirements, September 1999.

[9] V. Griffith and M. Jakobsson. Messin' with Texas: Deriving mothers maiden names using public records. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security (ACNS)*, pages 91–103. Springer-Verlag, 2005. LNCS no. 3531.

[10] RSA Security Inc. RSA SecurID authenticators, 2006. Product Specification. Referenced 2006 at www.rsasecurity.com.

[11] J. Jubak. Globalization isn't what's killing GM. *MSN Money*, 29 November 2005. Referenced 2006 at moneycentral.msn.com.

[12] D. V. Klein. Foiling the cracker: A survey of and improvements to, password security. In *UNIX Security II: USENIX Workshop Proceedings*, pages 5–14, Berkeley, CA, 1990.

[13] J. Leyden. Office workers give away passwords for a cheap pen. *The Register*, 18 April 2003. Referenced 2006 at www.theregister.co.uk.

[14] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, pages 110–124, 2005.

[15] K. D. Mitnick and W. L. Simon. *The Art of Deception: Controlling the Human Element of Security*. Wiley, 2002.

[16] T. Pullar-Strecker. NZ bank adds security online. *Sidney Morning Herald*, 8 November 2004. Referenced 2006 at www.smh.com.au.

[17] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger password authentication using browser extensions. In P. McDaniel, editor, *USENIX Security*, pages 17–32, 2005.