

Call Center Customer Verification by Query-Directed Passwords

Lawrence O'Gorman, Amit Bagga, Jon Bentley
Avaya Labs, Basking Ridge, NJ, USA

logorman@avaya.com, bagga@avaya.com, jbentley@avaya.com

Abstract

We introduce an authentication framework called Query-Directed Passwords (QDP) that is designed to incorporate the convenience of authentication by entrenched (long-term) knowledge questions (such as “What is your favorite marine animal?”), but to offer stronger security than from personal questions as traditionally used. Security is strengthened for this scheme by imposing several restrictions on the questions and answers, and specifying how QDP is implemented in conjunction with other factors. Four QDP implementations are examined for the call center application. We examine the security and convenience of one of these implementations in more detail. This implementation involves client-end storage of questions in a computer file or a wallet card, and follows a basic challenge-response authentication protocol.

Keywords: authentication, passwords, call center, verification, security.

Submitted to: Research Session category of Financial Cryptography 2004.

1. Introduction

“Good morning, Alice, and thank you for phoning ABC Corporation. For customer verification purposes, please tell me your: social security number, mother’s maiden name, date of birth, name of first pet, ...”

The amount of information in the request above may be exaggerated, but the scenario is not. We experience similar customer verification procedures at the beginning of many call center sessions, especially those dealing with health insurance and personal finances. Yet, the practice of using personal, entrenched knowledge (knowledge in our long-term memory) for authentication is perplexing. Is knowledge of Alice’s date of birth a good piece of information to use for security? If so, should Alice schedule her birthday party far away from the true birth date so that no one can learn this “secret” information? And should Alice give her social security number¹ to any call center agent who asks? Not only may she have privacy concerns with this, but if social security number is used for verification to many call centers, isn’t this like re-using a password for different hosts, which we are advised not to do?

It is clear why a password is not ideal for call center customer verification; the customer cannot remember yet another password. However, are the personal entrenched-knowledge questions good substitutes? Let’s compare the two. A password is a secret shared by a user and an authenticating host. In contrast, personal knowledge like your social security number and date of birth is not a shared secret and can be learned by an

¹ With apologies to countries whose national identification number is not called a social security number, we use this US number as an example of a unique identifier that is often requested for authentication. A similar apology can be given to those in countries where mother’s maiden name is a foreign concept.

attacker through some amount of research effort. A password can be changed if compromised. However, you can't change your mother's maiden name. With passwords, we are instructed to use different ones for different hosts to eliminate cross-attacks. However, there are only a few examples of personal knowledge that are traditionally used for authentication so the need to reuse them rises with each new host to which we register. Worse yet, because these personal-knowledge questions are known for particular hosts (e.g., host X always uses mother's maiden name), these become a standing target for an attacker to learn this information about customers. To avoid standing target questions, some verification procedures require the user to create questions of their own (e.g., "What is my first pet's name?"). However, there is a danger here that users may choose questions poorly. Choosing a poor question is similar to choosing a weak password that can be guessed or found by dictionary search. For these and other reasons, authentication by these personal-knowledge types of questions is not a good substitute for passwords. So, why is it still widely used? It is because memorized passwords are forgotten [1-4] and entrenched knowledge is not.

There are alternatives to knowledge-based authentication by password or entrenched knowledge [5]. Tokens, such as smart cards can store or create strong passwords without the need for memorization. However, for customer applications, there is a significant expenditure to provide tokens and readers to all customers. Biometrics suffers the same cost drawback to provide scanners (readers) to customers, and also has reliability and logistic drawbacks (e.g., if a fingerprint is compromised, how do you change it? [6, 7]).

Because of deficiencies with the alternatives, we look to knowledge-based authenticators for the call center customer application. As maligned as knowledge-based authentication may be due to the inconvenience of memorization, passwords actually have a number of good qualities. Like the biometric, you've always got your passwords, at least once they are memorized. Unlike tokens and biometrics, you don't need special readers for passwords because computer keyboards and telephone keypads are ubiquitous. Passwords, when used correctly, can have very strong security, much stronger than most biometrics, which succumb to false matches at a rate similar to the guessing rate of a 4- or 5-digit PIN [5]. And because no extra equipment is needed, passwords are inexpensive.

Because of memorization difficulties, our work focuses not on memorized passwords but on passwords based on entrenched knowledge that are elicited through questions. Although we have just criticized the traditional use of this type of authentication, our goal is to retain its inherent convenience and to improve upon its weak security. We do the latter task by creating a more rigorous framework for use of entrenched-knowledge questions, formalizing question design, and limiting question scope.

Attempts to make entrenched knowledge authentication stronger are not unique. Ellison et al. [8] propose a method, called *personal entropy*, to encrypt secrets or passwords via the answers to several entrenched-knowledge questions. They base this upon Shamir's secret-sharing scheme [9], also called a (t, n) -threshold scheme, where a secret is distributed into n shares of which at least t of these is needed to reconstruct the secret. The n shares are encrypted and decrypted using hashed functions of the entrenched-knowledge questions and answers. The emphasis in this work is to offer the user fault

tolerance by allowing her to forget the answers to some small number of questions, but still achieve successful authentication.

Frykholm and Juels [10] offer an approach, called *error-tolerant password recovery*, with the similar goal of deriving a strong password from a sequence of answers to entrenched-knowledge questions. One portion of the method is similar to the personal entropy method, distributing the answers to questions in a vector that is used for encryption and decryption. However this latter method achieves fault tolerance not by secret sharing but by using error-correcting codes in a scheme called *fuzzy commitment* [11]. The emphasis in this work is offering cryptographically strong security to defend against the computationally unbounded attacker. They describe an experiment on 9 users over a one-week period for their system-authored questions with open answers.

Our approach is similar to these previous works in at least two respects. We also use entrenched-knowledge questions and can achieve fault tolerance with multiple questions. However, there are also differences in our emphasis and approach. The previous papers emphasize the cryptographic underpinnings of the approaches, whereas our paper deals with details closer to the human end. One example of this is design and types of questions. Our approach deals only with system-guided questions and answers. Another difference is in the use of the approach. Whereas the previous papers presented schemes for password storage and recovery, our paper applies query-directed authentication to customer call center verification.

In Section 2, we introduce our method, called Query-Directed Passwords (QDP). We describe specifications for questions and answers that underlie this approach. In Section 3, we describe the call center application, beginning with requirements and then offering four implementations with different security and convenience characteristics. Security comparisons are made between implementations and we investigate one implementation, involving client-end storage of questions, in more detail. Conclusions and future work are discussed in Section 4.

2. Query-Directed Passwords (QDP)

2.1 Background

User authentication can be divided into three categories (Figure 1): knowledge-based (which includes passwords), possession-based (which includes tokens), and ID-based (which includes biometrics). In this paper, we further divide the knowledge-based category into “push” and “pull” passwords. The distinction is that a push password must be pushed into the user’s memory (memorized) at registration. A “pull” password need only be pulled from the user’s memory (recalled) since it is based on entrenched knowledge (knowledge residing in long-term memory or permastore); it does not need to be memorized. All “pull” passwords are hint- or query-directed, however within this category we make a distinction between the traditional type and our approach. We call the first type *traditional pull-type authentication*, the stereotypical example being mother’s maiden name. We call our approach *query-directed passwords*, or *QDP*. The distinction is not in an entirely different approach, but in the fact that QDP applies a

formalism that constrains the types of questions and the procedure by which these questions are used.

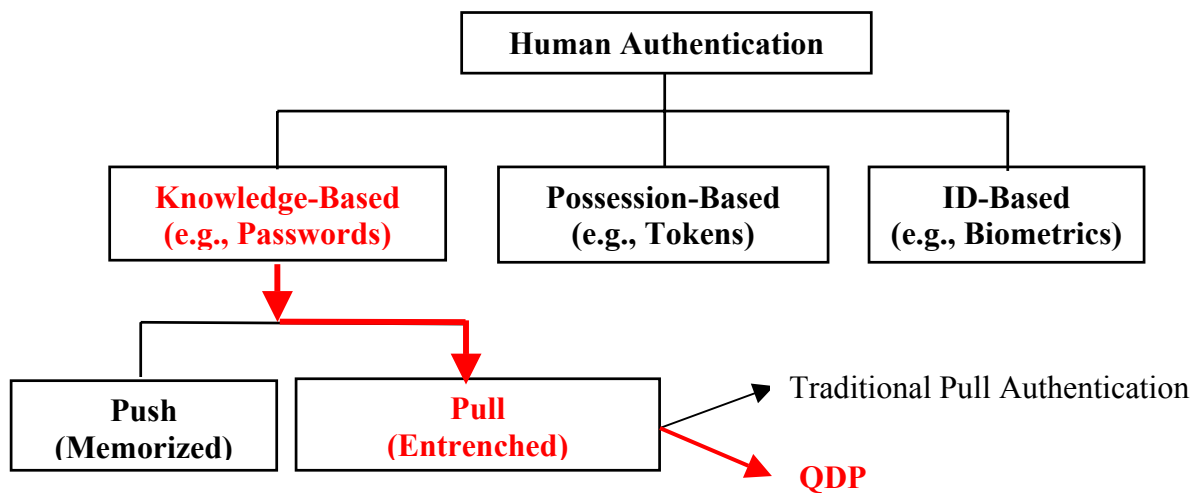


Figure 1 Hierarchy of human authentication, showing query-directed passwords under the category of knowledge-based, pull-type authentication.

We can describe three types of questions and answers for pull-type authentication. Examples are listed for each:

1. Open question and open answer (user creates the questions),
 - *What is the name of my first pet?*
2. Selectable question and open answer (system provides the questions for selection),
 - *What is my mother's maiden name?*
 - *What was the number of my childhood house?*
3. Selectable question and multiple-choice answer (system provides the questions and answers for selection),
 - *What is my favorite marine animal?*
1) whale, 2) shark, 3) dolphin, 4) seal, 5) sea horse, 6) minnow.
 - *Where was the car parked in my childhood home?*
1) right side, 2) left side, 3) front, 4) back, 5) garage, 6) street.

2.2 QDP Approach

In contrast to the traditional approaches of choosing a very limited number of personal questions or enabling the customer to create questions, we create QDP questions with strict guidelines. Instead of personal facts, QDP questions involve opinions or trivial facts such as, “What type of apple do you prefer?” and “Where do you usually carry your house keys?” These are unlikely to be on your resume or in official personal information, so are more difficult for an attacker to learn. Instead of a few static pieces of information, QDP requires the user to select a number of questions (e.g., 10 to 20). More questions helps authentication in three ways: 1) different questions can be used for different hosts,

2) different questions can be used for different authentication sessions for the same host, and 3) if a question becomes compromised, it can be eliminated with still other questions to take its place. Instead of the system choosing questions, some that the user might consider to be private, the system offers questions for user selection, so that any questions considered private need not be selected. Below, we list specifications of QDP answers and questions:

1. Answers must be consistently and easily recalled by a user over time.
2. Answers must be discriminating of a user. The answer space must be fairly evenly distributed across the population and individual answers must be independent of one another.
3. Answers must not be easily guessed or learned.
4. Answers must not be considered confidential.
5. Questions offered for user selection must be fairly large in number, and the question selection must be fairly evenly distributed across the population.
6. Questions chosen by a user must be dispersed in type or topic.

Comparisons between traditional questions and QDP questions are shown in Table 1.

Table 1 Comparison between traditional query-directed authentication and QDP.

Traditional	QDP
A few traditional questions, or user-created	System offers pre-created questions for selection
Personal facts that may be learned by others	Trivial facts and opinions, difficult to learn by others
Few questions	Many questions
Can't be changed if compromised	Eliminate question if compromised
Mandatory information may be considered private	User selects questions

Of the question and answer types described in Section 2.1, we restrict QDP usage to type 3 and a subset of type 2. The type 3, selectable question and multiple-choice answer, constitute the majority of a user's QDP questions. We call these *QDP multiple-choice questions*. It is straightforward to perform a security analysis upon this type because we know questions and permissible answers beforehand. Ideally, we also know population statistics such as frequencies of selected questions and answers that will help in security analysis.

QDP questions can also be a subset of type 2 questions, selectable questions and open answers, which involve particular questions with numeric answers. "What was the number of my childhood house?" is an example of this. We call these *QDP numeric questions*, and these are often used where a traditional PIN would be used, the advantage of the QDP numeric question being that it is elicited from entrenched memory with a query, so it is more easily recalled. In Section 4, we will briefly describe how we use information extraction techniques to guide a user toward a permissible question and answer of this type. The numeric restriction facilitates this analysis.

We do not use type 1 questions, open questions and open answers. This is because of the difficulty in automatically analyzing such questions and answers (the question would need to be understood by a computer as a precondition to security analysis).

Use of QDP multiple-choice questions alone does not provide the level of security required of most applications. If an authentication system were designed with 4 questions of 6 multiple-choice answers each, a brute force attacker would be able to guess the answer after $6^4 / 2 = 648$ guesses on average. If the attacker has some knowledge of the user or some knowledge of the distribution of answers, he could guess even more quickly. This is far less secure than for a 4-digit PIN, for which $10^4 / 2 = 5000$ is the average number of guesses needed. Therefore, the security story does not end with the QDP questions and answers. Our own use of QDP involves some other contributor to overall system security such as a second factor or a particular protocol to strengthen security. This is evident in the description of implementations in Section 3.

We have performed user testing of QDP and report this elsewhere [12]. One component of this testing was an experiment on answer recall over time. In a short-term test done weekly for 30 users over a 3-month period and longer term testing done 5-6 months after enrollment, users answered about 95% of individual questions correctly. With fault tolerance on the 5 to 8 questions asked per session, users successfully authenticated 98.5% of the attempts.

3. Call Center Application

QDP has properties that make it applicable to such tasks as password recovery and use of voiced password in a public area [12]. In this paper, we consider the application of call center authentication. Customer verification is common for financial- and health-related call centers. In this paper, we focus on *call* centers (implying just telephone communications) rather than *contact* centers (phone and Web communications). This is because the former presents a more constrained and challenging problem. The methods presented here apply to the superset of call center transactions as well.

3.1 Specifications

Security is the main objective in authentication, however an important consideration in specifying this application is that the authenticating person is a customer. Unlike employees, military personnel, or civil servants, there is little leverage to *make* customers do anything because they can switch to a competitor if they are dissatisfied. Therefore, a well-designed call center will offer both security *and* customer convenience. These specifications are listed below.

I) Security-Related Specifications:

1. It should be difficult for an attacker to authenticate in place of the authorized customer.
2. It should be difficult for potential attackers at the host-side to learn the customer's authentication information.
3. Compromise detection and recovery is desirable.

4. It should be difficult for an attacker to mount a successful denial of service attack.
5. Adjustable security to enable a range of security strengths should be possible.

II) Customer-Related Specifications:

1. It should be convenient with respect to time and memory effort for an authorized customer to authenticate.
2. The customer's privacy should be respected.
3. The customer should be allowed some fault tolerance to get a portion of the authentication response wrong but still to successfully authenticate.
4. Verification should be possible via the numeric keypad of a telephone.
5. Verification should also be possible by speaking into the telephone without worry that an eavesdropper can learn the authentication information.
6. The scheme should be inexpensive on a per customer basis.

We describe these specifications below and examine how these are met for particular QDP implementations in Section 3.3.

Security specification 1 is the main purpose here, to safeguard confidential information and to prevent transactions by unauthorized persons. There are two types of client-side attackers: strangers and acquaintances. Strangers do not know anything about the customer prior to the attack. They can guess answers or first learn about the customer via whatever public means are available. Acquaintances (which span the range from an office colleague to a spouse) already know some things about the customer. With knowledge of the customer an attacker can mount an imposter attack.

Security specification 2 refers to host-side or insider attack. For instance, if a call center operation allows agents to see customer access PINs, then a dishonest agent could steal these for fraudulent use. In the world of computer administration, user passwords are usually transformed and stored via a 1-way hash function. In this way, even administrators with access to password files cannot easily learn user passwords because there is no direct transformation from the hashed value back to the password.

Security specification 3 states that some means of compromise detection and recovery is desirable. A bankcard, for instance, provides tangible means for detecting one form of compromise – when you can't withdraw money because your card is gone, it may have been stolen. Recovery involves canceling the card and obtaining a new one. In contrast, personal data are almost in a permanent state of "compromise" because many people know your date of birth, etc. Furthermore, compromise recovery is inconvenient or impossible with data such as date of birth.

Security specification 4 states that denial of service attacks should be difficult. One way to cause the system to deny service to a true customer is for someone to answer the questions incorrectly a number of times. Most systems will freeze the account after a number of failed attempts to prevent brute force attacks.

Security specification 5 allows fewer questions to be asked for a low-security application, and more for stronger security. Furthermore, even within a single session, the customer

can start with a “default” security level, then be required to answer more questions if a transaction requiring higher than default level is requested.

Customer specification 1 states that authentication should be convenient. Security and convenience often present conflicting specifications, so a compromise must be chosen. Since authentication includes both enrollment and verification, one can anticipate another tradeoff between these. For instance, a longer time spent to enroll diligently might save time later each time the customer verifies.

Customer specification 2 regards privacy, what information a person wants to keep confidential. But personal privacy concerns are different for different people. The system should be able to handle differences in privacy preferences.

Customer specification 3 sets query-directed authentication apart from passwords and other authentication options. If you forget a password, you will not be authenticated. There is no middle ground. The same is true if you lose a smart card or your voice fails to verify. However, if a customer forgets an answer to one or more authentication questions, we’d like to offer him some leeway to still authenticate. It is important to say that strength of security should not suffer here, so if for example a customer misses an answer, tolerance is not defined as simply asking more questions until he gets one right. Correct and incorrect answers must both be considered.

Customer specification 4 is a practical requirement for the targeted telephone medium. Although it might be convenient to speak an authentication response, speech recognition technology is still not reliable enough to correctly recognize that response consistently, especially for unconstrained telephone speech (untrained speaker, unlimited vocabulary, random telephone line) [13], and for instances of high background noise. Therefore, interaction is assumed to be in the rudimentary mode of touch-tone key entry into an IVR (Interactive Voice Response) system. (This does not preclude speech interaction, but just stipulates that our methods must also work for the more constrained requirement of numeric key entry.)

Customer specification 5 allows the customer to speak authentication responses (in addition to keying them). However, the customer should not have to worry about an eavesdropper overhearing and using the information for attack. An example test of this is the following. A customer who works in an open office environment checks her financial information by phone each day. Specification 5 states that her office neighbors who can overhear these calls should not be able to learn information such as to subsequently impersonate her.

As many as possible of the adjectives used in the requirements above, “difficult,” “easy,” “fault tolerance,” and “security strength” should be quantified. Security and convenience are not always measurable in an absolute sense, however we will attempt to compare relative merits among options.

3.2 Implementations

We describe four ways to implement QDP for call center authentication (see Figure 2):

1. Host store – basic
2. Host store plus PIN.

3. Host store plus address.
4. Client store.

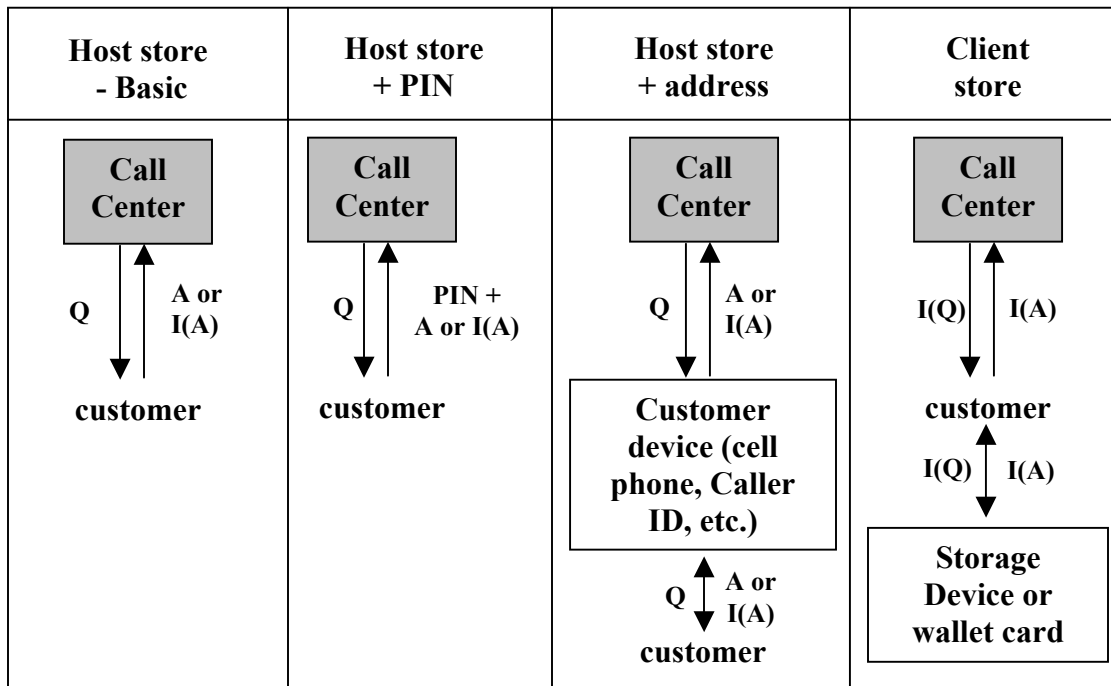


Figure 2 The four options for call center implementation are illustrated. Questions Q or question indices I(Q) are sent by the call center. Answers A or answer indices I(A) are returned.

3.2.1 Host Store – Basic

The basic host store implementation uses QDP alone in the procedure described in Section 2. This provides least security among implementations. The customer phones the call center and is connected with the IVR system for performing authentication. After identifying herself by keying in an identity number, the system asks a random subset of the customer’s enrolled questions. The customer responds by keying in the index of each multiple-choice answer. If all of the answers are correct (or some pre-set portion of them), then the customer is successfully authenticated.

There are a number of vulnerabilities with this basic implementation. One is that any attacker can learn some questions selected by the customer just by phoning the call center and entering that person’s (non-secret) identification number. After learning the questions, the attacker can make an effort to find the answers. So these questions become a standing target for an attacker. Another problem is the low entropy. For example, if there are $k=4$ questions with $M=6$ multiple-choice answers each, this results in a keyspace of $6^4=1296$, which is guessable in 648 tries on average. This is not very strong security. Finally, personnel at the authenticating host can easily learn a customer’s information. Questions and answers cannot be hashed as for computer passwords because (for one reason) a telephone has no computing power to perform this computation. They can be encrypted at the host, but are vulnerable when decrypted for use.

Notwithstanding these problems, the basic implementation of QDP is more secure than traditional pull-type authentication. For a client-end attacker to expose the customer's questions he could have the following strategy. Phone the call center to expose the k questions that are asked for a single authentication session; and answer these with random guesses, likely failing authentication. Endeavor to learn the answers to these k questions offline. Phone again and answer any previously exposed questions to which he has learned the answers, and expose new questions. Continue this process until authentication is successful. If the total number of questions a user has selected is n , then the minimum number of sessions to expose all questions is n/k . For $n=15$ and $k=4$, the attacker would need at least 4 sessions to expose all questions, and could successfully attack in the next session if he learned all the answers. We can prevent an attacker from being able to expose all the questions by freezing the account after some number of failed questions. If an attempted attack is suspected in this case, then the exposed questions can be retired from those selected by the customer. This cannot be done with traditional pull-type authentication, where the number of potential questions is either fixed or very small.

The basic QDP scheme can prevent host-end personnel from learning questions and answers from one system, then using them on another. This is because questions are not as limited or few with QDP as traditional pull-type authentication. A customer can use a different set of questions and answers for different hosts. In contrast, there are a limited number of hosts where only mother's maiden name, social security number, and date of birth can be used.

3.2.2 Host Store Plus PIN

A weakness of the basic implementation is that an attacker can learn the customer's questions merely by entering into an authentication session. We can defend against this vulnerability by asking for a PIN before relaying the questions. If the entered PIN is correct, the system asks k questions selected by the customer. If the entered PIN is incorrect, then the system still asks k questions, but these are not necessarily the customer's selected questions but rather are a random selection from the N questions in the QDP database. In this way, an attacker cannot first learn the PIN by brute force attack, and then independently expose the questions. By posing the selected questions depending upon the correct PIN, this increases the keyspace from 10^4 plus 1296 for a 4-digit PIN and for four questions, versus 10^4 times 1296, about 4 orders of magnitude greater.

The drawback of requiring a PIN is that the customer will now have to memorize something, and if she forgets will not be able to access her account. This is where a QDP numeric question can replace a traditional PIN for the convenience of the customer.

3.2.3 Host Store Plus Address

Instead of requiring the customer to use a PIN, we can keep the customer's selected questions confidential in another way. When the customer wants to authenticate, the questions are sent only to a previously registered communications device. If the customer registers the home phone number, then the caller identification can be used for this. The same is true for a cell phone or the address of a pager or wireless PDA. The degree of security depends on how easily an address is spoofed and how exclusively a customer

maintains the device. We have built an application where the questions are sent as a text message to a cellular phone. This has two advantages. One is that the questions are sent only to the registered personal device. The other is that a customer can visually read text far faster than waiting for the same questions to be spoken.

3.2.4 Client Store

There is an inherent drawback to the host store QDP implementations described above because they are vulnerable to host-side attack. An alternative to these is to store the questions not at the host but at the client.

The QDP enrollment procedure is the following. The customer accesses a QDP question database, likely via the Web. This database lists questions with multiple-choice answers. The questions have indices, $q_i, i = 1, \dots, N$ and the answers have indices, $a_j, j = 1, \dots, M$. The only peculiarity of this database is that question and answer indices are reordered in a random way for each enrollment session. That is, a particular question and particular answer will likely have different indices for different visits to the database. The customer does not identify herself, but selects questions and downloads them to a file on her computer or prints them out. She mentally chooses answers to her selected questions – but doesn't circle the answers or indicate these in any way. The last step of the enrollment is for the customer to send to the authenticating host a file that contains the question indices and the customer's chosen answer indices.

The verification procedure is straightforward. The customer identifies herself whereupon the authenticating host queries the customer with randomly chosen question indices from enrollment. The customer looks up in her file what questions correspond to those indices and answers with the indices of her answers. If the customer answers all correctly, or a number within some pre-determined tolerance, then she successfully authenticates.

The customer file is a rudimentary codebook. It contains a list of codewords, which are the question indices, and corresponding decoded "plaintext", which are the answer indices. However, only one of the M answer indices per question is correct. The true codebook owner possesses the "key" to this codebook by having knowledge of the answers to the questions. Therefore, to succeed at authentication, a person needs two things. One is the codebook and the other is the "key."

3.3 Security

3.3.1 Implementation Comparisons

Table 2 shows how well the different implementations of Section 3.2 meet specifications of Section 3.1.

The "host store – basic" implementation is the least taxing of the customer. If the user selects QDP questions that she can always remember, there is no extra effort needed. However, a major drawback of this implementation is that it has less resistance to client attack than the other implementations. This is because an attacker can learn the customer's questions by phoning repeatedly, then either learning or guessing answers to these questions.

Table 2 Shows how well different QDP call center implementations meet specifications listed in Section 3.1. A “Y” indicates that specification is met. An “N” indicates that specification is not met. A “–” indicates that specification is met under some circumstances and not under others.

	Host Basic	Host + PIN	Host + address	Client
Security Specifications:				
1 – resist client attack	–	Y	Y	Y
2 – resist host attack	N	N	Y	–
3a – offer compromise detection	N	N	–	Y
3b – offer compromise recovery	Y	Y	Y	Y
4 – resist denial of service attack	N	N	Y	Y
5 – offer adjustable security	N	Y	Y	Y
Customer Specifications:				
1 – convenient	Y	Y	–	–
2 – privacy respected	Y	Y	Y	Y
3 – fault tolerance	Y	Y	Y	Y
4 – numeric keypad input	Y	Y	Y	Y
5 – voice input without fear of eavesdroppers	Y	–	Y	Y
6 – inexpensive	Y	Y	–	Y

The “host store plus PIN” implementation requires the next least effort of the customer. A drawback of this implementation is that, if an eavesdropper can hear a customer’s PIN, he can then obtain questions and attack the system in the same way as for the basic implementation. We can strengthen the host plus PIN implementation by requiring the PIN to be entered via the keypad and by using a number of QDP numeric questions that are chosen randomly across authentication sessions.

The “host store plus address” implementation requires the customer to have some device, whether home phone with Caller ID or cell phone. There is expense involved if the customer does not otherwise have one of these, and this requirement is potentially inconvenient because the user must have the device when authenticating. This implementation meets all security specifications except compromise detection.

The “client store” implementation compares well among these implementations. It has similar security to the “host store plus address” implementation, but a major advantage is that it entails no device expense. Since this is a “what you know” and “what you have” combination, the customer will be inconvenienced if she forgets the paper or file on which the indices are written. It is vulnerable to a host attack because the host stores indices for each question and answer. However, indices can easily be changed periodically, and they are different for different hosts to reduce the danger of cross-host attack.

3.3.2 Client Store Security

The client-store implementation has some attractive properties of user convenience and security. We examine security of this scheme in more depth in this section.

The client store verification procedure follows a challenge-response protocol. The server sends a challenge vector, which is a set of k question indices. The customer looks up each question by its index, chooses an answer, and returns the index of the answer. The customer returns k answer indices in total. A straightforward way to number the answer indices is sequentially from 1 to M for each multiple-choice question. However, if this is done, the potential answer keyspace is only M^k . For example, for $M=6$ choices and $k=4$ questions, this is $6^4=1296$. A stronger security scheme is to use random numbers for the multiple-choice answer indices. Now, the answer keyspace can be arbitrarily high, with the tradeoff that the customer must enter longer numbers for each answer. For example, if each answer index is chosen randomly from the range $[0, \dots, 99]$, then the potential keyspace increases to $100^4=10^8$; if the range is $[0, \dots, 999]$, then the keyspace is $1000^4=10^{12}$.

If an attacker steals the wallet card containing the customer's selected questions, then vulnerability increases. In this case, using random number answer indices adds nothing to the security strength because the attacker knows the potential answer index for each question. For $M=6$ choices and $k=4$ questions, the keyspace is 1296. However, just like a physical house key, the wallet card provides evidence of compromise detection. When the customer finds it is missing, she should notify the call center of this, whereupon the current questions will be cancelled.

We use the term, "poor man's token" for the client store implementation in which a wallet card (or piece of paper) stores the questions and multiple-choice answers. Like using a more expensive challenge-response security token, a customer can receive a random number challenge and return the appropriate random number response. However, there is one significant difference between the electronic token and the paper one. The electronic token has a very large number of challenges that it can respond to. Challenge numbers are typically 8 to 10 digit numbers, so the number of challenges is 10^8 to 10^{10} . In contrast, a wallet card holds n questions of which k of these are used in a session. There are $n!/(n-k)!$ permutations of questions, which for $n=15$ and $k=4$ is 32,760. This is not a large number of challenges, but is likely to be large enough to defend against direct replay attacks given the low frequency of call center authentications per customer. The easier attack is if an eavesdropper can learn the discrete question indices and corresponding answer indices, then he can successfully impersonate the customer. The eavesdropper could learn all answer responses in as few as n/k sessions, or 4 sessions for our example. This attack can succeed if authentication is performed over a channel in plaintext. Telephone transmission is usually in plaintext, since the terminal has no computing power to perform encryption, however capture of an analog telephone transmission requires physical presence to tap the telephone line. In contrast, data network eavesdropping does not require physical presence. So, any transmission of authentication information via data channels should be encrypted. The SSL (Secure Sockets Layer) protocol is commonly used for contact center communications on the Web.

4. Conclusions and Future Work

We can conclude from the implementations proposed and from the results of preliminary user testing reported elsewhere [12] that the QDP approach to call center customer verification is promising. However, since much of this work is for the purpose of improving customer convenience, the real proof of this will only be found when the scheme is used in a full-scale system. We are in the process of implementing a version of the client store implementation for password recovery use within Avaya. This application has a call center component where users can phone a number to obtain aid in recovering their password.

There are a number of areas that require more investigation for this approach. One involves statistics of the QDP multiple-choice questions. Specification 2 in Section 2.2 listed uniformity of the question and answer spaces and independence of answers to be important for discriminating among users. We have done smaller-scale tests on these statistics showing a fairly good spread of selections [12]. However, no question or answer spread will be perfectly uniform, so we are investigating different weighting schemes to identify and correct for questions and answers that may give an attacker advantage due to non-uniform selection.

Another area of current work is important for the QDP numerical questions. Since the answers are open, we need a proactive method (such as in [14]) for recognizing “good” or “poor” answers. For example, the question, “What is my current telephone number?” is a poor question because an attacker can easily learn the answer. However, the question, “What is Fido’s telephone number?” is a better question because an attacker would have difficulty associating a dog’s name with the owner’s telephone number. We are working on a method that employs information extraction techniques to search for question and answer associations in telephone directories and in general web searches.

There are other open issues that will become clearer with further testing, longer experience, and use in real applications. Memory recall has only been tested over 5 months. Will users still recall their answers consistently over longer periods of time? Our current database of 200 questions was authored without any particular expertise in the psychology of question design and users’ responses. Can questions be better designed? And, no security proposal is complete until opened to the scrutiny of security experts and the abuse of potential attackers. Will the QDP approach offer the combination of better convenience along with stronger security as proposed?

5. Summary

This paper investigates call center customer verification using an authentication scheme involving entrenched-knowledge questions, called Query-Directed Passwords (QDP). QDP is shown to differ from traditional entrenched-knowledge authentication schemes (e.g., “What is your mother’s maiden name?”) due to a QDP framework that adheres to strict requirements on the types, number, and use of questions. This framework provides stronger security while maintaining the inherent convenience of the traditional approach. Four implementations for call center customer verification are described here. These four implementations differ in how and where the QDP questions are stored. One of the most secure implementations involves client-side storage of the questions on a computer file or

simply on a wallet-card. This offers the stronger security of a challenge-response protocol along with other advantages such as compromise detection and low cost.

References

1. R. Morris, K. Thompson, "Password security: A case history," *Comm. ACM*, Vol. 22, no. 11, Nov. 1979, pp. 594-597.
2. D.C. Feldmeier and P.R. Karn, "UNIX password security – ten years later," *Advances in Cryptology – CRYPTO '89 Proceedings*, Springer-Verlag, 1990, pp. 44-63.
3. S. M. Furnell, P. S. Dowland, H. M. Illingworth, P. L. Reynolds, "Authentication and supervision: A survey of user attitudes," *Computers and Security*, Vol. 19, no. 6, 2000, pp. 529-539.
4. J. Yan, A. Blackwell, R. Anderson, A. Grant, "The memorability and security of passwords – some empirical results," TR 500, University of Cambridge, Computer Laboratory, September 2000, <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-500.pdf>
5. L. O’Gorman, "Comparing Passwords, Tokens, and Biometrics for User Authentication," *Proc. IEEE*, submitted for publication, 2003.
6. L. O’Gorman, "Seven issues with human authentication technologies," *IEEE Workshop on Automatic Identification Advanced Technologies*, Tarrytown, New York, March 2002, pp. 185-186.
7. C. Dorai, N. K. Ratha, R. Bolle, "Dynamic behavior analysis in compressed fingerprint videos," *IEEE Trans. Circuits and Systems for Video Technology*, Special Issue on Video-Based Biometrics, Oct. 2003.
8. C. Ellison, C. Hall, R. Milbert, B. Schneier, "Protecting secret keys with personal entropy," *J. of Future Generation Computer Systems*, 16 (4), Feb. 2000, pp. 311-318.
9. A. Shamir, "How to share a secret," *Communications of the ACM*, Vol. 22, No. 11, Nov. 1979, pp. 612-613.
10. N. Frykholm, A. Juels, "Error-tolerant password recovery," in P. Samarati, ed., *Eighth ACM Conference on Computer and Communications Security*, ACM Press. 2001, pp. 1-8.
11. A. Juels, M. Wattenberg, "A fuzzy commitment scheme," in G. Tsudik, ed., *Sixth ACM Conference on Computer and Communications Security*, ACM Press, 1999, pp. 28-36.
12. L. O’Gorman, A. Bagga, J. Bentley, "Revisiting Query-Directed Authentication," (submitted for publication), 2003.
13. J. Fiscus, W. M. Fisher, A. Martin, M. Przybocki, D. S. Pallett, "NIST Evaluation of Conversational Speech Recognition over the Telephone," *Speech Transcription Workshop*, Maryland, May, 2000. <http://www.nist.gov/speech/publications/>
14. M. Bishop, D. V. Klein, "Improving system security via proactive password checking," *Computers and Security*, Vol. 14, no. 3, 1995, pp. 233-249.