Proceedings of the IEEE, Vol. 91, No. 12, Dec. 2003, pp. 2019-2040 © 2003 IEEE

Comparing Passwords, Tokens, and Biometrics for User Authentication

Lawrence O'Gorman
Avaya Labs, Basking Ridge, NJ, USA
logorman@avaya.com

Abstract

For decades, the password has been the standard means for user authentication on computers. However, as users are required to remember more, longer, and changing passwords, it is evident that a more convenient and secure solution to user authentication is necessary. This paper examines passwords, security tokens, and biometrics – which we collectively call authenticators – and compares these authenticators and their combinations. We examine effectiveness against several attacks and suitability for particular security specifications such as compromise detection and non-repudiation. Examples of authenticator combinations and protocols are described to show tradeoffs and solutions that meet chosen, practical requirements. The paper endeavors to offer a comprehensive picture of user authentication solutions for the purposes of evaluating options for use and identifying deficiencies requiring further research.

Keywords: end-user authentication, human authentication, access control, verification, identity management, password, identity token, biometric

1. Introduction

In times gone by, authentication was not a complex task. One person, call her Alice, would meet another person, Bob, and either recognize him by visual appearance, or not. If Alice did not recognize Bob, he could explain that he was a friend of a friend, or a business envoy, etc., and Alice could decide whether to believe him. Of course, if Alice and Bob were spies, they would use more formal methods for mutual authentication – from piecing together two halves of a ripped page to exchanging pre-arranged nonsense statements [1]. But spies were the exception.

Enter the computer era and authentication has changed. Now we cannot "see" the entity on the remote end of a computer network, and indeed the entity could be a friend, a machine, or an attacker. We exchange information about our finances and health that we wish to remain as private as any spy correspondence. The World Wide Web adds a new complication since attackers can access our records without the need for physical presence. Whether it is for protection of our own records or our own digital identities, we have been forced to adopt more formal authentication methods even in our common lives. Pass phrases, identity tokens and biometrics are no longer just the domain of spies. We now use these authentication methods routinely in our interactions with computers and over computer networks. For this purpose, it is important to understand the authentication options, how effective they are, and how they compare.

Authentication is the process of positively verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to allowing access to resources in the

system [2]. The authenticating entity accomplishes positive verification by matching some short-form indicator of identity, such as a shared secret that has been pre-arranged during enrollment or registration for authorized users. This is done for the purpose of performing trusted communications between parties for computing and telecommunications applications.

In this paper, we differentiate between *machine-by-machine authentication* (or simply *machine authentication*) and *human-by-machine authentication* (*user authentication*). (See Figure 1.) The former includes well-established protocols that can be very secure. An example is the Secure Sockets Layer (SSL) protocol that is employed when making secure Internet transactions [3] (and is often indicated by the appearance of a locked padlock on your Internet browser). However, machine authentication simply verifies machine identities and gives no assurance of the identity of the person at the machine. This is the job of user authentication. Therefore, we can more narrowly define user authentication as the process of verifying the validity of a claimed user.

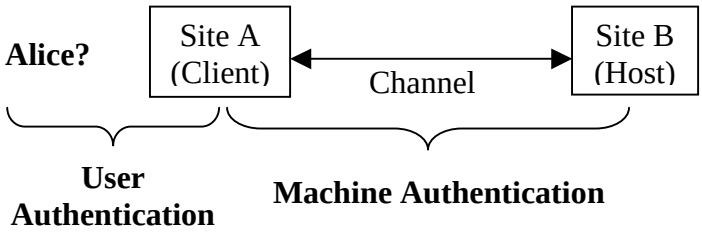


Figure 1 Authentication comprises user authentication between human and machine, and machine authentication between machines. Sites A and B can authenticate each other, but user authentication asks, is it really Alice at Site A?

Although user authentication has been practiced far longer than computers and telephones have been in existence, it is much less secure than machine authentication. Consider, for instance, the Advanced Encryption Standard (AES) that has recently been adopted as the standard encryption algorithm for the US government [4]. In physical terms, this algorithm is like a very strong bank vault, practically impossible to break into. For AES, the user chooses a private key to perform encryption and decryption. For the vault there is a combination. The maximum AES key length is 256 bits. If an attacker were to try to guess the key, it would require on average over 10^{76} guesses to do so, too time-consuming even by computers in the foreseeable future. However, a 256-bit key is too long for most humans to remember, so in practice this key is stored in a computer file protected by a more memorable password. Herein lies the problem, because humans often choose a password that is not only memorable to them, but also easily guessable by a person or computer [5-12]. Using the bank vault analogy, this is like storing the vault combination on a piece of paper in a hidden place close to the vault. Now, all an attacker has to do is to find the piece of paper, and use the combination to open the vault. The strongest vault can be attacked by exploiting a human mistake, just as the strongest encryption algorithm can be attacked by exploiting a weak password. Because user authentication deals with humans, complete with our limitations and foibles, and because it often is the front-end protection of otherwise strongly secure systems, it is variously called the “Achilles heel,” the “weak link,” and the “last yard” of secure systems.

The focus of this paper is a comparison of human authenticators. Comparison factors are security, convenience, and cost. The latter two factors are relatively straightforward and are described only briefly in this paper; however security as measured by vulnerability to applicable attacks is not so straightforward and thus constitutes the bulk of the paper. For a broader description of the field of user authentication, see [13]. For sources of information on individual authenticators, see Chapter 9 of [13] for security tokens, [14, 15, 16] for biometrics, and any of several security texts such as [13, 17-19] for passwords.

This paper is organized as follows. Section 2 gives a background introduction to user authentication, including definitions of authenticator types, security terms associated with user authentication, biometric concepts, and compatibility issues. In Section 3, we discuss comparison factors for authenticators. These factors are used as the basis for comparing authenticators, enabling one to choose the most appropriate authenticator for an application. In Section 4, we examine relative authenticator strengths against a pertinent list of attacks and security issues. In Section 5, we discuss choosing appropriate authenticators for particular applications. Finally, we conclude in Section 6 with general recommendations of where and how authenticators are most appropriate.

2. Authenticator Background

This section provides an introduction to authenticators and related security matters. Terminology and concepts that are introduced in this section will be used throughout the paper.

2.1 Authenticator Definitions

We use the term *password* to include single words, phrases, and PINs (personal identification numbers) that are closely kept secrets used for authentication. There are many studies showing the vulnerabilities of password-based authentication schemes [5-12]. The basic problem with passwords can be explained succinctly: a memorable password can often be guessed or searched by an attacker and a long, random, changing password is difficult to remember.

An *identity token*, *security token*, *access token*, or simply *token*, is a physical device that performs or aids authentication. This can be a secure storage device containing passwords, such as a bankcard, remote garage door opener, or smart card. This can also be an active device that yields *one-time passcodes*, either *time-synchronous* (changing in synchrony with a master at the host) [20] or *challenge-response* (responding to a one-time challenge). Token security defenses include tamper-resistant packaging and special hardware that disables the token if it is tampered with or if the number of failed authentication attempts exceeds a chosen threshold. When we refer to “token” in this paper, the general concept will be a portable, secure storage device accessed at the client end via a password to obtain a passcode that is transmitted to the host for authentication. A *passcode* is a secret number like a password, except it is machine-generated and machine-stored, so it can be longer, more random, and perhaps changing.

A *biometric* is a feature measured from the human body that is distinguishing enough to be used for user authentication. Biometrics include: fingerprints, eye (iris and retina),

face, hand, voice, and signature, as well as other more obscure or futuristic biometrics [14, 15] such as gait and smell. A biometric purports to inextricably link the authenticator to its owner, something passwords and tokens cannot do, since they can be lent or stolen. When used to verify the person involved in a transaction, an inextricable link can offer the property of *non-repudiation*. This property provides proof of a transaction such that the involved parties cannot subsequently reject the transaction as unauthorized or disclaim having participated in it. However, biometric features can be copied or counterfeited – with varying levels of difficulty – and used to gain unauthorized access to a security system [21-23]. So even biometrics cannot offer a guaranteed defense against repudiation, as will be discussed further in Section 4.7. This paper takes into account these issues to compare authenticators and their combinations.

2.2 Security Definitions

Security systems and methods are often described as *strong* or *weak*. When used in relative terms, the meanings are clear. A door with a lock offers stronger security than one with no lock. A credit card number alone offers "weak" defense against repudiation because a user can easily deny a credit card charge by claiming that his credit card number was stolen. However, a credit card number *plus* a signature has "strong" defense (meaning "stronger" defense than without a signature) because the user leaves evidence of his presence by his signature.

It is more difficult to measure security in absolute terms. One way to measure absolute strength and weakness of security systems is as follows. A strong system is one in which the cost of attack is greater than the potential gain to the attacker. Conversely, a weak system is one where the cost of attack is less than the potential gain. Cost of attack should take into account not only money, but also time, potential for criminal punishment, etc.

In Section 4, we describe the strengths and weaknesses of authentication features versus given attacks. For instance, a token can offer *strong* defense against brute force guessing (because it can store or create a number much longer than a memorized number and thus incur less risk of being guessed randomly). However, it is *weak* in defending against theft. Since we don't presume any particular application, and therefore cannot measure the cost of attack or potential gain, these are not absolute measurements. Instead, they are relative to other methods. So, for the token example, "strong defense" against guessing should be read as "stronger defense than most other methods described here." And "weak defense" should be read as "weaker than most other methods described here." One purpose of using these relative descriptions is to identify authenticator combinations that complement strengths and reduce weaknesses against different attacks.

A caveat that should be stated is that a user can always use an authenticator poorly so as to make a "strong" authenticator "weak". When these terms are used for comparisons in this paper, we assume that the authenticator is being used as recommended to attain the best security for which the authenticator is capable.

In this paper, we apply *authentication* narrowly to focus on *remote computer authentication* (as opposed to authenticating to a standalone PC or to a human gatekeeper). Figure 2 illustrates two schemes for remote computer authentication. Scheme 1 involves direct authentication through a network channel to a host. This

includes the common procedure of sending a password to the host where the submitted password is compared against the stored password for the claimed user. Scheme 1 also includes submission of a biometric through a reader at the client machine, where the biometric is *not* matched, but is sent to the host for matching. Scheme 2 involves an authenticating intermediary, of which there can be two options. For scheme 2a, the user submits an authenticator to an intermediary, which in turn sends a passcode to the authentication server. The intermediary might be a token, or a biometric matcher, or client-end, password storage/retrieval software. The user first authenticates to this intermediary, then the intermediary sends out a passcode to the host. Alternatively, for scheme 2b, the intermediary may be a single sign-on server at the host. In this case, the user has only one authenticator, but the service can authenticate to multiple hosts by sending the password or passcode from secure storage. In either of the scheme 2 options, the point of the intermediary is to increase security (long passcode from shorter password) or convenience (multiple passcodes from a single password), or both.

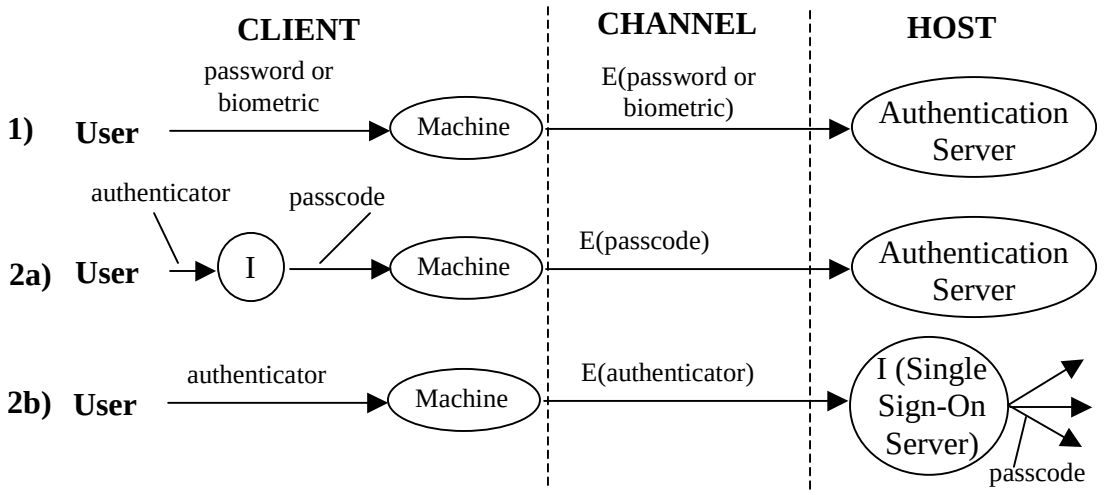


Figure 2 Schemes for remote authentication. 1) User submits password or biometric template through client machine to host machine for authentication. 2a) User authenticates to intermediary I, at the client (such as a token, biometric matcher, or password storage program), and a passcode is sent to host indicating the result of that authentication. 2b) User submits authenticator through client to intermediary single sign-on server, from which point an appropriate password or passcode is sent to one of multiple hosts. In the channel, E() designates that the transmitted message is sent encrypted.

Authenticators can be attacked at three locations: at the client, in the transmission channel, and at the host. Other papers cover protection of a password or passcode in the channel by protocols that encrypt the password [24-26]. We deal in this paper only with security issues at the client and host.

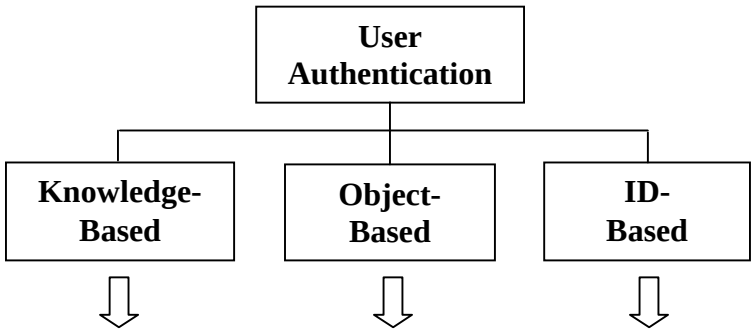
2.3 Types of Authenticators

Authentication factors are usually grouped into these three categories: 1) what you know (e.g., password), 2) what you have (e.g., token), and 3) who you are (e.g., biometric). This is a good mnemonic scheme and unlikely to fall from use, but it is not without problems. For instance, a password is not strictly *known*; it is memorized. Implying

otherwise risks minimizing a major problem with passwords, forgetting them. Biometrics are definitely not "who you are" any more than hair color or body build indicates your true self. A biometric is simply one feature of your appearance. We prefer the following authenticator labels: knowledge-based, object-based, and ID-based. These are described below and illustrated in Figure 3:

1. **Knowledge-Based** ("what you know"**)** – are characterized by secrecy or obscurity. This type includes the memorized password. It can also include information that is not so much secret as it is "obscure," which can be loosely defined as "secret from most people." Mother's maiden name and your favorite color are examples in this category. A security drawback of secrets is that, each time it is shared for authentication, it becomes less secret.
2. **Object-Based** ("what you have") – are characterized by physical possession. Physical keys – which we call metal keys to distinguish them from cryptographic keys – are tokens that have stood the test of time well. A security drawback of a metal house key is that, if lost, it enables its finder to enter the house. This is why many digital tokens combine another factor, an associated password to protect a lost or stolen token. There is a distinct advantage of a physical object used as an authenticator; if lost, the owner sees evidence of this and can act accordingly.
3. **ID-Based** ("who you are") – are characterized by uniqueness[1] to one person. A driver's license, passport, credit card, university diploma, etc., all belong in this category. So does a biometric, such as a fingerprint, eye scan, voiceprint, or signature. For both ID documents and biometrics, the dominant security defense is that they are difficult to copy or forge. However, if a biometric is compromised or a document is lost, they are not as easily replaceable as passwords or tokens.

[1] An ID-based authenticator is intended to be unique. For an ID document such as a driver's license, it is one document for one person. We avoid the question of whether a biometric has "one in the world" uniqueness, and instead claim that it is distinctive to the degree that it is highly unlikely that two biometric authenticators will be exactly alike, at least within the scope of a particular implementation. For more on uniqueness of biometrics, see [27].



Commonly Referred to as:		Password, Secret	Token	Biometric
Support Authentication by:		Secrecy or obscurity	Possession	Uniqueness and personalization
Security Defense:		Closely kept	Closely held	Forge-resistant
Example	**Traditional:**	Combination lock	Metal key	Driver's license
	Digital:	Computer password	Key-less car entry	Fingerprint
Security Drawback:		Less secret with each use	Insecure if lost	Difficult to replace

Figure 3 User authentication is split into three authenticator categories. Attributes of these are listed.

Note that biometrics fall into the ID authenticator category and their security does not depend on secrecy. Face and voice are obviously not secret, and it is difficult to keep a fingerprint or iris secret from a determined attacker. A biometric is like a number on a driver's license – it is not the secrecy of the number that makes it a good authenticator; it is the difficulty to counterfeit the original "document" [28]. (For more on the secrecy of biometrics, see [29, 30].)

Different types of authenticators can be combined to enhance security (see Table 1). This is called *multi-factor authentication*. For security purposes, each authenticator result must be satisfied; in effect a Boolean AND operation is performed for each factor's authentication results so all must be affirmative. A common example of multi-factor authentication is the bankcard. The combination of a bankcard plus a password – two-factor authentication – is a better choice than a card alone because the card can be stolen and used, whereas a card that is password-protected cannot be used without knowing the secret. This example of token plus password constitutes the vast majority of current multi-factor implementations. If a password is difficult for the user to remember, a biometric ID can protect a token alternatively, but this usually entails higher equipment cost than a password. Password and biometric ID are not often combined because biometrics are usually included for the sake of convenience, to avoid having to remember a password. Generally, multi-factor authentication that combines all three factors has not been widely applied, although some high security applications may require this.

Table 1 Combining authenticators provides security advantages and can increase or decrease convenience.

Authenticator Combination	Security Advantage	Convenience Drawback	Example
Knowledge- and Object-Based	Lost/stolen token protected by password	Must carry token and memorize password	PIN-enabled bank card
Object- and ID-Based	Lost/stolen token protected by ID	Must carry token, but not ID if it is a biometric	Photo-ID
Knowledge- and ID-Based	Two factors provide security in case either compromised	Have to memorize password and have ID.	Password and biometric for computer access.
Knowledge-, Object-, and ID-Based	A third factor to provide security in case two other factors are compromised	Have to memorize password, carry token, and have ID.	Military applications requiring photo-ID checked by guard, plus password.

2.4 Biometric Types

Biometrics differ from the other authenticators in ways that are described here. Biometrics are usually classified as physical or behavioral types. The physical type includes biometrics based on stable body features, such as fingerprint, face, iris, and hand. The behavioral type includes learned movements such as handwritten signature, keyboard dynamics (typing), and gait. Speech is usually categorized as behavioral because it is a product of learned behavior; however the underlying body feature upon which speech is based is the vocal apparatus (lungs, vocal cords, nasal tract, vocal tract), which is physical and relatively stable. In fact all biometrics used for authentication depend to some degree upon a physical body feature; otherwise there is no constant upon which to authenticate. Due to these ambiguities, we suggest a different classification that doesn't involve the physical and behavioral labels.

Instead of classifying the biometric itself, we classify the *biometric signal* that we measure. There are two types (see Figure 4):

1. Stable biometric signal.
2. Alterable biometric signal.

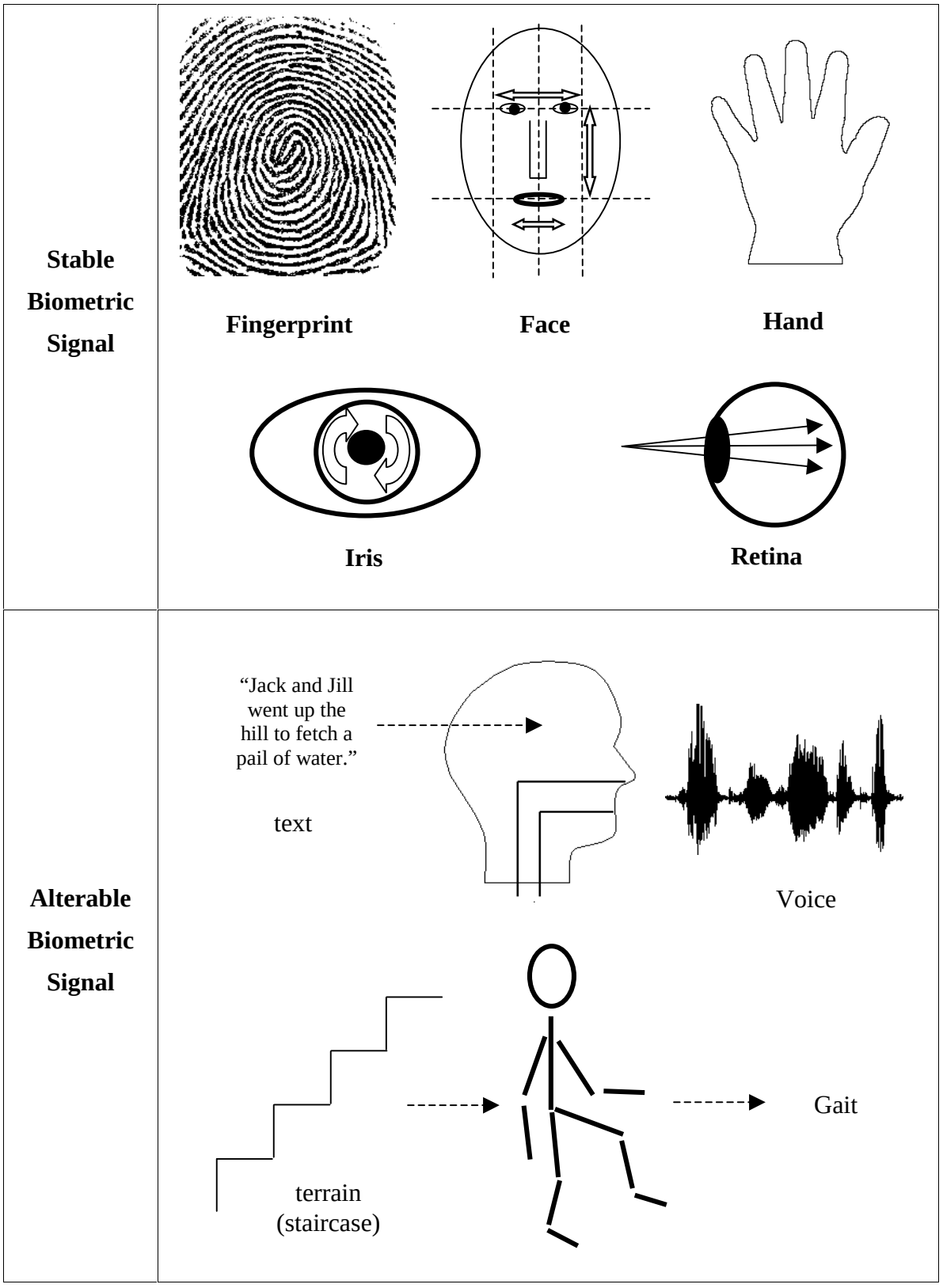


Figure 4 Examples of stable and alterable biometric signals.

A *stable biometric signal* is relatively constant in time. Except for minor perturbations due to noise (and excluding drastic obfuscation by accident or plastic surgery), the features used for matching stabilize before or at maturity. Biometric matching is usually not done on the raw signal. Instead, a smaller-sized template of these features is first extracted. For a stable biometric signal, the biometric template, *BT*, is determined directly from the biometric signal, *BS*, which is acquired directly from the biometric, *B*. That is, $B \rightarrow BS \rightarrow BT$. Therefore the template for the stable biometric signal (designated by subscript "*S*") is a function simply of the unchanging biometric,

$$BT_S = f(B).$$

For example, a fingerprint image is a biometric signal, *BS*, and the extracted minutiae features constitute a biometric template, BT_S. Since *BT* is directly extracted from *B*, this is an example of a stable biometric signal.

In contrast, an *alterable biometric signal* is comprised of two components, the underlying, stable biometric, *B*, and a variable, *x*. These are combined to yield the signal, *BS(x)*, from which is derived the template, *BT(x)*. That is, $\{B, x\} \rightarrow BS(x) \rightarrow BT(x)$. Therefore, the template for an alterable biometric signal (designated by subscript "*A*") is a function of a stable component and a variable,

$$BT_A(x) = f(B, x).$$

For example, a speech signal, *BS(x)*, is the result of vocalization of a variable, *x* (word or phrase), through the stable vocal tract filter, *B*, and the feature set extracted from this is the template, $BT_A(x)$. Similarly, for the handwriting biometric, the variable is text, and for the gait biometric the variable is a combination of terrain and tempo.

It is not true that fingerprint, face, eye, and hand are always stable biometric signals, and that voice, handwriting, and gait are always alterable biometric signals. For instance, one could devise an alterable face biometric signal that measures the shape and extent of facial feature movement as a sentence was spoken or an emotion displayed. One could also devise an alterable eye biometric that includes measurement of pupil reaction to light. We can go the other way as well. Consider a speaker verification scheme in which the user is asked to vocalize a particular vowel at a particular tone. In this case, the difference in speakers is due to their vocal tracts exclusively – there is no variable component. This is a stable biometric signal from speech.

For verification, an alterable biometric signal can be matched in either of two ways. The complete signal, $BT_A(x)$, can be matched. Or, the signal can be separated into its components, and these matched,

$$BT_A(x) = f(B, x) \rightarrow \{B, x\}.$$

For the speaker verification example, the *x* component could be a secret that undergoes speech recognition and is matched with the secret in the host's password file. *B* would describe the speaker. This is an example of 2-factor authentication: password and biometric.

Why do we make this distinction between stable and alterable signals? An alterable biometric signal can be an active component of a challenge-response protocol. Challenge-response protocols are powerful tools of secure authentication, as will be discussed in

Section 3.4. Conversely, stable biometrics cannot respond to a challenge – they are always the same. See Appendix 1 for a discussion of other limitations associated with stable biometric signals.

For the majority of this paper, when we use the term "biometrics," this means a stable biometric signal or either type (where the difference is not pertinent). When alterable biometric signal is the topic (as it is in Section 3.4, case 4; and Section 5.6), we specify this by using the full term.

2.5 Biometric Error

A user can forget or mistype a password, or can lose a token. These errors are inconvenient but the user has only himself to blame. Far more frustrating is system error where the user is not at fault and is unable to remedy the problem. Although computers can go down, keypads can malfunction, and token readers can fail to read, the rate of hardware error is low compared to errors of some biometrics, which can reject the user up to a few times each 100 attempts (see Table A2-1). Biometric error can occur for several reasons. The capture device might be dirty. The lighting might be poor. The system might have initially made a poor enrollment decision. The system might not adjust well to different environmental factors (cold, rain, sun glare, dryness, etc.) or to day-to-day variability of users.

There are two types of biometric error: verification error and identification error. Verification error describes error for a biometric system in which an attempt is made to match against a single identity (1-to-1 matching). We describe verification error for a biometric system by the error rate pair:

- $FNMR_k$: experimentally determined k-attempt false non-match rate
- *FMR(1)*: experimentally determined single-attempt false match rate[2].

The parenthesized "1" indicates verification against a single user. For a cooperative user in a verification system, $FNMR_k$ measures user inconvenience due to erroneous rejection. *FMR(1)* indicates system vulnerability due to an attacker being able to impersonate an authorized user.

Identification error describes error for a biometric system in which an attempt is made to match one person in a database containing records of that person plus many others (1-to-many matching). We describe identification error by the error rate pair:

- *FNMR*: experimentally determined false non-match rate
- *FMR(N):* experimentally and analytically determined false match rate for matching against a database containing N samples

In an identification system, *FNMR* measures the vulnerability of the system due to not identifying a true match in the database. An example of this is a face identification

[2] We should properly use $FMR_k(1)$ instead of *FMR(1)* because k-trial false match rate will be larger than for a single verification attempt. However this is not analytically calculable (because the trials are different but not independent) and usually not tested. So, we give the benefit of the doubt to the biometric system and use single-trial false match rate even though it is associated with k-trial false non-match rate.

system that fails to recognize a criminal even though his face is in the database. (Identification usually does not involve multiple attempts as for verification, thus no subscript, *k*.) *FMR(N)* measures user inconvenience of being misidentified in the database. An example is a system that identifies an innocent person as belonging to a criminal database. Assuming independence among biometric samples[3], the *FMR(N)* is calculated as,

$$FMR(N) = 1.0 - [1.0 - FMR(1)]^N . \qquad (3)$$

Therefore, false match rate for an identification system depends on the number of samples in the database. One can see from the equation that the probability of a query sample matching one or more of the *N* samples in the database increases logarithmically until the limit of 1 is reached when $N \to \infty$. Therefore, the probability of false match for identification is greater than that for verification and it increases with database size, *N*. (Further details on the statistics of biometric matching can be found in references [14, 15, 31, 32].)

To understand the magnitude of biometric error, we include FMR and FNMR data for various biometric modalities in Appendix 2. One should judge this data only after reading all the descriptions in that Appendix. This data shows that FNMR is in the range of 1-2% for multiple authentication attempts of fingerprint, voice, and hand biometrics; 5-10% for face biometrics; and about 0.25% for iris. FMR is in the range of 0.01-0.15% for fingerprint, hand, and voice; 5-10% for face; and 0.0001% for iris.

2.6 Compatibility with the Underlying Authentication System

In this section, we describe how user authentication fits into full authentication systems. The point of this section is that the choice of authenticator will be influenced by the current computing infrastructure; not all authenticators will be compatible. We describe compatibility with respect to three authentication protocols: RADIUS, Kerberos, and (generic) single sign-on.

The first protocol we describe is RADIUS (Remote Authentication Dial In User Service) [33, 34]. Companies, universities, etc., use RADIUS software for managing identities of users requesting access to computing resources that may involve a number of networked machines. This protocol involves a shared, centralized authentication server (called the RADIUS server) upon which all users' authentication data are stored. User requests for remote access may be made to one of many machines (called RADIUS clients), but these machines relay requests to the single RADIUS server. At this server, the request is evaluated and an authentication result passed back to the RADIUS client, then to the user.

[3] The validity of statistical independence among biometric samples should be qualified. Stable biometrics are treated as independent samples for different subjects. This is based on experimental evidence that extends over 100 years for fingerprints and 20 to 40 years for iris, face, and hand. Alterable biometric signals as defined in Section 2.4 may not be independent. If the variable component of an alterable biometric signal is fixed across subjects (e.g., the same utterance is spoken by different subjects for speaker verification), then there is a degree of dependence across signals due to that common variable. However, the stable biometric component of the alterable biometric signal is in fact the component that is used to distinguish among subjects. To the degree that this stable biometric component can be separated from the variable component, we assume this to be statistically independent for different subjects as for the other stable biometrics. More detailed treatment of biometric independence is found in [32].

The RADIUS protocol supports the conventional static password that is stored in a secure manner (MD5 hash function [35]). It also supports one-time passcodes that are generated at the time of request for some token and smart card authentication methods – that is, they are not read from storage as for the static password. It is important to emphasize that RADIUS handles password and token authentication differently; furthermore, despite its wide use, it does not handle all authenticators.

Another authentication protocol is Kerberos [36, 37]. This is a popular network authentication protocol based on cryptographic key distribution [38]. For a human user (the system also facilitates machine authentication), initial authentication is made to Kerberos in a conventional fashion such as with login and password. However, this is done only once (per session). Upon successful authentication, "tickets" are issued to the user enabling her to prove her identity and gain authenticated access to various resources. This is done transparently. The user also receives a session key to encrypt and decrypt messages to defend against eavesdropping and replay attacks and to safeguard message integrity even over unprotected networks. Kerberos is the standard network authentication option for user verification in Windows 2000. The current version of this operating system supports password and smart card authenticators. Compatibility of other or non-standard authenticators cannot be assumed.

A final authentication example is single sign-on (SSO). This enables a user's single authentication action to a server to provide access to connected computers and systems to which she has access permission without the need to reenter passwords. SSO can be used for an employee to enter a corporate computer system by logging into one machine with a password, and then having access to other machines on the corporate network without further authentication. This same approach can be used for Internet access [39]. The user authenticates to a single site, and then the SSO server handles authentication to subsequent sites. One protocol upon which SSO is built is Kerberos. Another is SAML (Security Assertion Markup Language), an XML-based framework for exchanging security information [40]. From the user's perspective, SSO reduces the number of passwords she is required to memorize. Although this reduces the burden, most users will still have to remember multiple passwords since it is unlikely that all authentication tasks will be serviced by a single SSO service.

Virtually all authentication protocols accept the traditional password, although there will be different rules on password length and character set. Many standard systems also accept some time-synchronous and challenge-response tokens, however compatibility cannot be assumed for all tokens. Biometrics is less widely compatible than the other two authenticator types at this time.

3. Comparison Factors

There are several factors by which we compare authenticators. These are described in this section.

3.1 Keyspace and Entropy

Keyspace is defined as the range of different possible values of a key. A password with n characters, where each of those characters can have c different values will have a keyspace size of,

$$k_p = c^n. \quad (1)$$

Statistical *entropy* is a measure of variation or uncertainty [41]. This is measured in bits. The password keyspace size, k_p, relates directly to the maximum entropy, H_{max}, for randomly chosen authenticator numbers,

$$H_{max} = \log_2 k_p \text{ [bits]}.$$

The pertinent difference between keyspace and entropy is that the former is an absolute measure of maximum or best-case, whereas the latter is statistically related to how users select from the keyspace. Take 4-digit PINs for example. The keyspace size is 10^4 = 10,000. That is, there are a maximum of 10,000 different PIN choices. If PINs were generated randomly (with uniform probability over the entire keyspace), an attacker would have a 1 in 10,000 chance that any single guess would match a given PIN. The entropy of this is $\log_2 10000 = 13.3$ bits. However, if a user is allowed to choose her own 4-digit PIN, the keyspace remains the same, but the entropy can be much lower. This is because many users would choose a PIN that is more memorable than a random one. Say users chose a calendar date for their PIN in "ddmm" format. The first digit would have possibilities 0, 1, 2, or 3. The second digit would have possibilities from 0 to 9. The third digit would have possibilities 0 or 1 and the fourth digit 0 to 9. Therefore a PIN chosen in this way would have only about $4\times10\times2\times10 = 800$ possible values. Assuming these dates are chosen uniformly, the entropy is $\log_2 800 = 9.6$ bits, which is almost 4 bits fewer than maximum for the keyspace.

It is straightforward to understand that keyspace and entropy should be high enough to reduce the probability of successful guessing and brute force attacks. However, it is not always the case that a high keyspace system is more secure than a lower one. This is because system considerations are also involved. For instance, network authentication involving passwords is often limited to a few (e.g., 3-5) failed attempts before system *lockout*, in which case further authentication attempts are rejected. So a guessing attack at the client-side is unlikely to succeed even if the password has low entropy. Similarly, since a token usually employs two-factor authentication, a low-entropy, 4-digit PIN can be sufficient, since any attacker would have to steal the token as well. For authentication involving a physical action such as reading a smart card or scanning a biometric, each authentication attempt may take a second or so. Many fewer brute force attack attempts can be made for these two-factor cases as compared to an attack on only a password. A computer program can attempt millions of passwords per second, so the password alone would require much higher entropy than these two-factor cases (other considerations being equal).

3.2 Effective Keyspace of a Biometric

A biometric doesn't have a fixed number of possible values. Theoretically, the keyspace of biometrics such as fingerprints is unlimited because if you could measure the

continuous signal with infinite precision, no two would be the same. But, one could say the same for passwords, that if you allowed the password length to be unlimited, you'd also have an unlimited keyspace. For both passwords and biometrics, practical concerns limit the keyspace. In practice, a biometric is measured not in continuous space, but discretely. Furthermore, the discrete features are usually afforded a tolerance, so this means that the matching precision for a biometric is even lower than its sampling precision.

For comparison purposes, we can define the effective keyspace of a biometric. This is determined as follows. If the password keyspace is uniformly distributed, the probability of correctly guessing any single password sample is one over the keyspace,

$$P(\text{correct guess}) = 1 / k_p.$$

For a biometric, the probability of falsely matching is analogous to the probability of succeeding in a brute force password guessing attack. Given a biometric (such as the biometric of an attacker), the probability of it matching any other single biometric sample in a database is the false match rate for a single verification attempt, *FMR(1),*

$$P(\text{false match}) = FMR(1).$$

Since *P(*false match*)* for a biometric is analogous to *P(*correct guess*)* of a password, then *FMR(1)* is analogous to $1 / k_p$. So, we define the *effective keyspace* of a biometric as,

$$k_b = 1 / FMR(1). \qquad (2)$$

One has to be careful in comparing k_p and k_b. The k_b is based on an experimentally determined value of *FMR(1).* The k_p will be comparable only if the password character selection is randomly chosen.

3.3 Host-Side Security

Static passcodes are stored at the host for matching against passwords or passcodes submitted from the client. A passcode can be stored at the host in one of three forms:

1. Plaintext,
2. Disguised by reversible operation,
3. Disguised by irreversible operation.

The problem with storing a secret in plaintext is that it is no longer a secret to the host. It is readily readable by the host administrators, and its secrecy beyond the host is entirely dependent upon how securely the host maintains it. Hosts can be untrustworthy, administrators can be unethical, and files containing plaintext authenticators can be stolen. Plaintext storage is a poor way to store an authenticator.

A better way is to disguise the authenticators using a reversible operation such as encryption. This way, if the authenticator file is stolen from the host, the passcode is not directly readable. The thief needs to steal the decryption key as well as the file to reveal the plaintext. Although this increases the required effort of a thief, it does not defend against untrustworthy hosts or unethical administrators who have the key.

The authenticator can be safeguarded against host-side attack by using an irreversible operation, called a *1-way hash function*, or simply a *hash function* (use of the term in this paper is restricted to 1-way hash functions) [42]. A hash function takes a variable-length message and converts it to a fixed-length string or hash code (e.g., there are 160 bits for the common Secure Hash Algorithm (SHA) [43]). A good hash function for security use, often called a *cryptographic hash function*, has two important properties: 1) it is computationally infeasible to determine from a given hash code an input that maps to this output, and 2) it is computationally infeasible to find for a given hash code, a second input that maps to this same output (for a 160-bit hash function, the degree of difficulty is $2^{80} = 1.2\times10^{24}$ operations). Consider a plaintext password, *P*. When this is operated upon by a hash function, *h()*, the result is,

$$P \rightarrow h(P).$$

For authentication, the host needs only to maintain the hash function and the hash value of a password. When the user wishes to authenticate, the host sends the hash function to the client, the user enters a password, *P'*, this is hashed, and the result is returned to the host. The host compares this response against its copy in storage,

$$h(P') =? h(P),$$

(where "*=?*" designates the match operation whose result can be "yes" they match or "no" they do not match.) Therefore, proof of authentication can be established without host-side knowledge of the user's password.

Host storage for biometrics is different than for passwords. There is little need to store biometrics secretly – from the security standpoint – since we stated in Section 2.3 that biometrics are not secret at their origin. However, for privacy reasons, it is often desired that stored biometrics be protected [29]. Hashing is not an option. This is because biometrics are matched not exactly but by "closeness", and hashed numbers do not maintain the property of closeness. Instead, biometrics are stored at the host as encrypted templates, an encrypted vector of matching features whose file is usually much smaller than the original biometric signal.[4] To emphasize the parallel to a password stored at the host without hashing, we refer to a biometric being stored in *plaintext/template* form.

3.4 Authentication Protocols

The *challenge-response protocol* is a fundamental tool of secure authentication. This is a process that verifies an identity by requiring correct authentication information to be provided in response to an unpredictable challenge [45]. The challenge is usually a

[4] Besides using one-way hash functions to safeguard privacy of the original before hashing, hash functions are also used for memory-efficient comparison. For instance, a multi-page document can be hashed to a 160-bit word and stored. Then a document that is claimed to have the same content as the original can be hashed via the same function and its 160-bit word compared against the original to test equivalence. Since biometrics cannot be hashed, compression via hashing is not an option. However, biometrics are usually not stored in raw form, but instead as feature templates (except for law enforcement purposes, where they are stored as original signals or under lossless compression [41]).

random number[5], and the response is related to this number. Use of this protocol prevents an attacker from replaying a previous authentication response.

Below, we describe basic protocols for passwords, tokens, and biometrics. This is to show how each authenticator can participate in a challenge-response protocol and how the authenticator information is stored at the host. Although the protocol we describe in Case 1 is the basis for such widely used password protocols as Unix [5] and Windows NT and 2000 [46, 47] login, the actual protocols are generally more complex. We omit the complexity here to focus on how authenticators are involved.

Case 1: Password Protocol – The basic password challenge-response protocol is initiated when a user sends user identification, *U*, to the host in step 1. (See Figure 5.) In step 2, the host returns a random number, *r*, that will identify the session, a hash function, *h()*, and a challenge function, *f()*. In step 3, the user returns the response, comprised of the result of the function involving the hash of a submitted password, *h(P')*, and the submitted random number, *r'*. In step 4, authentication is granted if this result is equivalent to the result of the function with random number and the hash of the true user password, *h(P(U))*; otherwise it is not granted. Note that the user password, *P(U)*, is not stored in plaintext on the host; instead it is hashed to form *h(P(U))* to avoid theft at the host.

Case 2: Token Protocol – In the basic token authentication protocol, the token either stores a static passcode or generates a one-time passcode. (See Figure 6.) This is similar to the password protocol, however instead of a potentially weak password, a long and random passcode is first hashed, *h(W'))*, combined with the random number challenge, and then transmitted as the response to the host. The user accesses the passcode from token storage with a password, *P'*, but that password is used only between the user and the user-held token. The user passcode can be stored in hashed form at the host, *h(W(U))*, or it can be generated for one-time passcodes. *A*uthentication of the password at the token can be done similarly to Case 1.

The following two cases involve biometric matching. Case 3 pertains to a stable biometric signal or to an alterable biometric signal that does not take advantage of its alterability to engage in a challenge-response protocol. Case 4 describes a challenge-response protocol that can only involve alterable biometrics.

Case 3: Stable Biometric Protocol – This is a basic challenge-response protocol for a stable biometric that is matched at the host. (See Figure 7.) A biometric, *B'*, is captured and processed on a biometric device at the client to obtain a biometric template, *BT'*. This template is combined with the random number challenge, *r'*, then encrypted, *E()*, and returned as the response to be matched at the host. In Figure 7, we also show a rudimentary procedure for authentication of the capture device where the device returns its identification, *D'*, that is compared with a list of registered devices at the host database, *{D}*.

[5] A *nonce* is a more general term for the random number challenge that is generated by the host in a challenge-response protocol A nonce is used to prevent replay of the transaction, and can be a time stamp, a sequential visit counter, or a random number. For simplicity in this paper, we use random number with the understanding that other nonce types may also be appropriate.

The basic challenge-response protocol for a stable biometric that is matched at the client is similar to that matched at the host. The distinction is that a biometric is captured, processed to a template, *BT'*, and matched to yield a yes/no match result, *BM'*, all at the client. The information is transmitted to the host, which determines authentication depending on a correct match and the legitimacy of the biometric device. The host contains no biometric information; instead the biometric template is stored at the client.

Case 4: Alterable Biometric Protocol – This is a basic challenge-response protocol for an alterable biometric signal that is matched at the host. (See Figure 8.) One difference from the stable biometric signal is that we can now involve the actual biometric in challenge-response, whereas we could not before. To do this, a challenge, *x*, is sent from the host to the client. This challenge is a random sequence of numbers, characters, or words. This is much shorter than the random number, *r*, because the user will have to vocalize it (speaker verification), type it (keyboard dynamics verification), or write it (handwriting verification) to yield the biometric signal, *BS'(x')*. This response is returned to the host, where processing is done to extract *x'* and *B'*. The recognized *x'* is compared with the challenge originally sent, *x*. The biometric, *B'*, is compared with that in the database corresponding to the user, *B(U)*. If *B'* matches *B(U)* and if *r'* matches *r*, then authentication is successful. Note a difference here from the stable biometric protocol is that the capture device need not be machine-authenticated. There is no need to do this here since the challenge-response protocol defends against replay and forgery, and matching is performed at the host.

The basic challenge-response protocol for an alterable biometric signal that is matched at the client is similar to that matched at the host. The distinction is that a biometric is captured, processed to a template, *BT'*, and matched to yield a yes/no match result, *BM'*, all at the client. The result is sent to the host along with a device identifier to verify that it is registered and unmodified. As compared with host matching, this protocol saves transmission bandwidth and template storage space at the host, at the cost of a more powerful and trustworthy device at the client.

	Client	Transmission	Host
1	*U*, user,	$U \rightarrow$	
2		$\leftarrow \{r, h(), f()\}$	*r*, random num, *h(), f()*,functions,
3	*P'*, password; *r'*, random num	$f(r', h(P')) \rightarrow$	
4		← yes/no	If $f(r', h(P')) = f(r, h(P(U)))$ then yes; else no

Figure 5 Basic challenge-response protocol for a password. (Case 1)

	Client	Transmission	Host
1	*U*, user	$U \rightarrow$	
2		$\leftarrow \{r, h(), f()\}$	*r*, random num; *h(), f()*, functions
3	$P' \rightarrow W'$ password to passcode via token; *r'*, random num	$f(r', h(W')) \rightarrow$	
4		← yes/no	If $f(r', h(W')) = f(r, h(W(U)))$ then yes; else no

Figure 6 Basic challenge-response protocol for a token. (Case 2)

	Client	Transmission	Host
1	*U*, user	$U \rightarrow$	
2		$\leftarrow \{r, E()\}$	*r*, random num; *E()*, function
3	$B' \rightarrow BT'$, biometric; *D'*, biometric device; *r'*, random num	$E(r', D', BT') \rightarrow$	$E^{-1}(r', D', BT')$ $= \{r', D', BT'\}$
4		← yes/no	If $r' = r$ and $D' \in \{D\}$ and $BT' = BT(U)$ then yes; else no

Figure 7 Basic challenge-response protocol for stable biometric. (Case 3)

	Client	Transmission	Host
1	*U* , user	$U \rightarrow$	
2		$\leftarrow \{r, x, E()\}$	*r*, random num.; *x*, random seq. challenge; *E()* functions
3	$B', x' \rightarrow BS'(x')$; *r'*, random num	$E(r', BS'(x')) \rightarrow$	$E^{-1}(r', BS'(x')) = \{r', BS'(x')\}$, $BS'(x') \rightarrow BT'(x') = f(B', x')$. Recognize *x'* from *BS'(x')*, Extract *B'* from *BT'(x')*
4		← yes/no	If $r' = r$, and $x' = x$, and $B' = B(U)$, then yes; else no

Figure 8 Basic challenge-response protocol for alterable biometric. (Case 4)

3.5 Convenience and Cost

If an authenticator is inconvenient, it won't be used, or won't be used properly, which may present vulnerabilities. Users who must remember multiple, changing passwords are notorious for abusing password rules. Though a token reduces the problem of remembering passwords, the user must remember to carry the physical object, which is sometimes inconvenient. Biometrics alleviates the problem of remembering anything, but some users experience inconvenience by false non-match results.

For tokens and biometrics in a networked application, there is an additional convenience issue of how to best register/enroll, renew, recover, and revoke the authenticator. Since a

token is an object, it must be put into the hands of the authorized person either personally or by delivery. Correspondingly, it may need to be removed from the user if authorization is revoked.

The tolerable cost of an authentication system is dependent upon the application. As mentioned in Section 2.2, one way to quantify this is to estimate the cost of the minimum-security implementation that makes the cost of attack to the attacker more than his maximum potential gain. However, this gambles that the attacker is fiscally rational. It is better to estimate the cost of loss to the attacked party and implement security to reduce the risk of successful attack to a chosen low probability.

There are three types of cost. One is the per-user cost. A password scheme costs nothing per user (if the user has a keyboard or keypad), whereas a biometric requires a reader at the client, and a token requires a reader and the token itself. Infrastructure costs can be large but are usually reduced on a per-client basis if that number is high. This is in contrast to the third cost, administration. Administrative costs (for example, for reset when a password is forgotten or token is lost) may be the most important consideration. These require ongoing expenditure for a trained labor force, the size of which increases with the number of users. A convenient authenticator reduces administrative costs.

4. Security Comparisons

We compare authenticators with respect to security issues in this section. Table 2 lists a number of potential attacks against user authentication with examples and typical defenses. Table 3 does the same for non-attack security issues. The following subsections expand upon the issues presented in these tables.

4.1 Client Attack

A fundamental property of good authenticators is that they should not easily succumb to *guessing attacks* or *exhaustive search attacks*. A large keyspace is desirable to defend against these types of attacks. It is straightforward to compare authenticators by keyspace. From equation (1), a 4-digit PIN has keyspace 10^4. An 8-character password whose characters are taken from the alphanumeric character set of 62 has keyspace equal to $62^8 = 2.2\times10^{14}$. However, humans don't usually choose within this keyspace efficiently (uniformly), instead tending more to dictionary and dictionary-derived words with a keyspace on the order of 10^6 to 10^7. (There are over half a million words in the Oxford English Dictionary [48].) A token can have arbitrarily high keyspace since human memory is not the limiter. Twelve digits are common, giving a keyspace of 10^{12}. From equation (2) and using the CESG results of Table A2-1, the effective keyspace of a fingerprint is $1 / 0.0001 = 10^4$, of an iris scan is $1 / 0.000001 = 10^6$, and of a face image is $1 / 0.16 = 6.25$. Comparing these, one can see that,

Token (10^{12}) > Password (10^{14}-10^6) > Iris (10^6) > Fingerprint, PIN (10^4) > Face (6.25).

When we limit the number of erroneous attempts before lockout at the client, all except for the face result are more than adequate keyspace sizes to defend at the client end.

A token is a good tool to generate high-entropy passcodes from lower entropy passwords and biometrics. In conjunction with a second factor, it can defend against search attacks

in general. The only requirement of the user is that the authenticator to the token, whether password, PIN, or biometric, cannot be easily guessed.

The biometric equivalent of trying to guess a password is trying to force a biometric system into a *false match*. This can be attempted by applying a biometric that is similar to the target of attack (such as a face of similar appearance). Or, a group of people can attempt a limited brute force search attack. For example, ten people can apply their one hundred different fingerprints to a system to increase the chance of a false acceptance.

4.2 Host Attack

Limited-attempt, random guessing is not likely to be successful at the client end even with low-entropy passwords. However, there is another reason to have a high-entropy password. This is to defend against an attack at the host end on the file in which the passcodes are stored. This can happen if the file is stolen or if an administrator with access to the file is untrustworthy. The most straightforward attack is a *plaintext attack* – if the passcodes are readable at the host they can be stolen. Credit card numbers are sometimes stolen this way. However password files are often stored in hashed form to prevent this attack. One can still attack a hashed file by performing a *dictionary search attack*, where words and combinations of words are hashed and then compared against hashed passwords for matches. An *exhaustive search attack* could also be tried, but will be too time-consuming for well-chosen passwords. An augmented defense to hashing is to add a few random bits to each hashed password, called *salt* [19]. This substantially increases the dictionary attack search time.

In a similar manner to mounting a plaintext search attack on a password at the host, a *plaintext/template attack* can be mounted against a biometric template stored at the host. However, because a biometric is not a secret, protection at the host is somewhat moot. The better protection against host vulnerability is to authenticate the capture device and for that device to assure that a biometric has truly been captured rather than entered as a file.

4.3 Eavesdropping, Theft, and Copying Attacks

Besides guessing, the next best low-technology way to learn a password is to steal it. This could happen by *eavesdropping* or by finding a piece of paper on which the password is written. Physical presence is necessary for these attacks and this limits the opportunity for attackers. A two-factor token is a good defense because it requires that the attacker needs to steal both the password and the physical token.

A token distinguishes itself from the other authenticators by the fact that it is a physical device. As such, it is susceptible to *theft* and *copying* (i.e., manufacturing of a counterfeit device). Physical possession provides much of the security of a token, much like a metal key. Unlike a metal key, there are additional safeguards. These include tamper-resistance, content encryption, and requirement of an additional factor to activate the token in case of theft or loss.

Analogous to the theft of a token is the *forgery* of a biometric (also known as copying, counterfeiting, or spoofing). Just as the authenticity of an ID document is dependent upon verifying its legitimacy at the point of acceptance, defense to this attack entails a liveness

or anti-forgery check at the biometric capture point [49]. As mentioned, the security of biometrics, or any ID-based authenticator, cannot rely on secrecy, but instead on the difficulty of replicating it.

4.4 Replay Attack

The *replay attack* can be considered a complement of the theft/copying attack. Whereas theft/copying involves an attacker obtaining the authenticator before entry at the client, a replay attack involves the attacker obtaining the authenticator in the channel between client and host. (See Figure 2.) Even if the channel signal is encrypted, as we have assumed in Figure 2, an attacker could circumvent the client capture stage and insert the encrypted authenticator into the channel. A challenge-response protocol defends against this attack. Because the challenge is session-specific and because the response incorporates the challenge inextricably, theft of a response for future replay attack would be fruitless outside of that session.

If a biometric is sent in plaintext/template form rather than combined in a response, then the biometric can be replayed. One defends against replay of a biometric by using a capture device that verifies the legitimacy of the biometric. To assure that an attacker has not replaced or altered a copy-detecting capture device, the device should participate in a secure machine authentication procedure with the host.

4.5 Trojan Horse Attack

A *Trojan horse attack* entails a rogue application masquerading as a trusted application for gaining information from, or entry to, a system. For authentication, this attack can be used to steal a password, token passcode, or biometric signal. The defense against this entails some assurance that the authenticator capture device (keyboard and computer for a password, token, or biometric capture device) can be trusted as legitimate.

An example of a hardware Trojan horse is a bank machine placed not by a legitimate bank but by attackers to learn customer card and PIN information. There is not much that can be done if a user decides to enter a static password, passcode, or biometric to an unknown machine that turns out to be malicious. Once stolen, the authenticator can be used in a legitimate machine. However, a token that generates a one-time passcode will not succumb to this type of attack, since one session’s passcode is useless in another session.

A biometric capture device could be replaced by one containing a Trojan horse. Consider a rogue fingerprint capture device that delivered a “yes-match” to anyone applying her finger to the device. This is why, when a decision is made at the client, the device must be machine-authenticated (see Case 3, Section 3.4, where client-side matching is discussed).

4.6 Denial of Service Attack

One drawback to limiting the number of authentication attempts is that an attacker can easily succeed at a *denial of service attack* by trying a false authenticator the requisite number of times to cause lockout. A defense to this is multi-factor authentication, in particular combining a token with a password or biometric. In this case, the attacker

cannot simply make *k* incorrect authentication attempts without first stealing the token (whose theft would result in denial of service as well, however token theft requires added effort by the attacker).

4.7 Non-Repudiation

Repudiation differs from the previously described attacks in that the legitimate user turns attacker upon the authenticating host, and n*on-repudiation* (defined in Section 2.1) is the defense against this. There are few good technical defenses against repudiation. However, a policy defense that makes the owner personally liable for all use of his authenticators, whether legitimate or not, would be effective (though draconian) at eliminating repudiation of credit card charges, for instance, since there would be no reason to deny charges if you were held responsible for them anyway.

A biometric offers non-repudiation to the extent that the capture device or the system effectively defends against theft, forgery, replay, and Trojan horse attacks. Furthermore, if matching is performed at the client end, then the capture device must be authenticated to the host.

4.8 Compromise Detection

Security defenses should not stop at resistance to front line – or first line of defense – attacks only. *Intrusion detection* methods attempt to recognize when illicit access has already been made into a security perimeter. In the context of authentication, we use the term, compromise detection. *Compromise detection* determines if an authenticator has been stolen or otherwise compromised, preferably before it is used illicitly.

Compromise detection mechanisms for passwords and biometrics are relatively weak, relying on the user to recall the last login date, for instance. For tokens, the tried-and-true method of compromise detection is observation of physical loss: when you lose your metal keys, you have physical evidence of this. A token provides this same physical indication of loss. When a token can be incorporated into a device that the user relies upon each day, such as a cell phone or watch, this increases the likelihood of effective compromise detection.

One notable difference between biometrics and other authenticators is that there is no option of compromise recovery for most biometrics. This is because a stable biometric signal cannot be changed. The only response to compromise detection is to revert to a password, because a compromised biometric should never be used again. The exception is an alterable biometric signal engaged in a challenge-response protocol (see Section 5.6).

4.9 Administrative and Policy Issues

There are also *administrative and policy issues* concerning *registration/ enrollment, reset, recovery, and revocation*. The main concern here is performing the operation only for the authorized person. It is important that the level of security required to perform any of these tasks be as great as or greater than the security level of the primary authenticator. For instance, if the secondary authenticator required for password reset is something as

weak as mother's maiden name, then this provides an easier target for attack than the primary password itself.

Table 2 Some potential attacks, susceptible authenticators, and typical defenses.

Attacks	Authenticators	Examples	Typical Defenses
Client Attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts; theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host Attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection (by administrator password or encryption) of password database
	Token	Passcode theft	1-time passcode per session
	Biometric	Template theft	Capture device authentication
Eaves-Dropping, Theft and Copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multi-factor authentication
	Token	Theft, counterfeiting hardware	Multi-factor authentication; tamper resistant/evident hardware token
	Biometric	Copying (spoofing) biometric	Copy-detection at capture device and capture device authentication
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode per session
	Biometric	Replay stolen biometric template response	Copy-detection at capture device and capture device authentication via challenge-response protocol
Trojan Horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device; client or capture device within trusted security perimeter
Denial of Service	Password, token, biometric	Lockout by multiple failed authentications	Multi-factor with token

Table 3 Other security issues in addition to attacks of Table 2.

Security Issues	Authenticators	Examples	Typical Defenses
Non-repudiation	Password, token	Claim lost or stolen authenticator	Personal liability, two-factor with biometric (e.g., signature)
	Biometric	Claim copied biometric	Capture device authentication
Compromise Detection	Password, biometric	Stolen password or copied biometric	"Last login" displayed to user to detect anomaly
	Token	Lost or stolen token	User notes physical absence
Administrative and Policy – Registration/ Enrollment	Password	Initial password registration	Delivery to pre-established e-mail address
	Token	New token registration	Delivery to pre-established, physical address
	Biometric	Biometric enrollment	In-person with picture ID
Administrative and Policy – Reset and Recovery	Password	Forgotten password	Secondary authenticator (e.g., date of birth)
	Token	Lost token	Delivery to pre-established, physical address
	Biometric	Compromised biometric	Not much option but to revert to password

5. Examples

We show a general procedure for building a security system in Figure 9. Material in this paper can help in a few of these steps. For the first step of risk assessment, Section 4 and Tables 2 and 3, describe some attacks and other risk issues related to authentication. In the next step, which includes the task of technically specifying the system, Section 3 and Tables 1, 2 and 3, describe different authentication system options and their specifications, advantages, and disadvantages. If biometrics are being considered, Table A2-1 can be used to get a notion of comparative recognition performance. In the implementation phase, the protocols described in Section 3.4 can form the basis of implemented protocols.

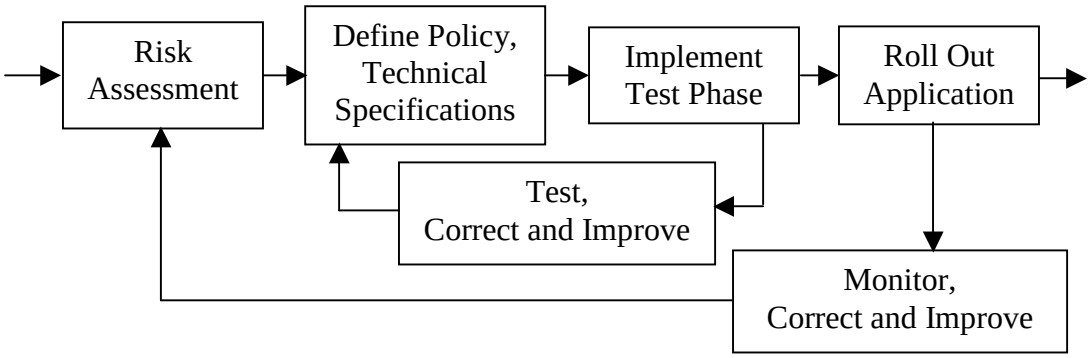


Figure 9 General procedure for building a security system.

Below are some examples of using material in this paper to choose among authentication options.

5.1 Authenticating Online (Network) Access

The password has been the standard for computer network access for decades. When used properly, it meets many requirements of this task. However, there are some drawbacks. Since we limit failed attempts before lockout for networked applications, it can succumb to a denial of service attack. Since it can be lent, it doesn't defend against repudiation. A password offers little compromise detection. Administration is easiest among authenticators, but that ease can lead to insecurity. Registration, reset, and recovery depend upon secure procedures, but these are often weak (e.g., dependent upon knowledge of mother's maiden name). Revocation is straightforward. It is convenient if the user is required to remember one or a few passwords, but inconvenient for too many passwords. It is relatively low cost.

To improve compromise detection (and convenience in the case of multiple passwords), a password and token combination has stronger security than a password alone. The penalty is increased cost for the token (token, reader, and system software) and the inconvenience of carrying it. The user still has to remember one password for the token, and this may be a burden if he has other passwords to remember. If this is the case, he can opt for a biometric-secured token. This latter option also offers better evidence against repudiation.

5.2 Authenticating Offline (Non-Network) Access

Examples of off-line access are: logging into a stand-alone PC or opening an encrypted file. Security policy often differs between offline access and online access. For online access, the number of failed attempts before lockout is often limited and the user is forced to contact an administrator to reset his password. Alternatively, a logging function keeps track of access attempts and any anomaly such as multiple failed attempts can trigger an alarm to which an administrator should react. Offline access often forgoes these safeguards, if there is no administrator at network end to reset the account. In this case, an attacker working offline can mount a brute force attack of a very large number of guesses.

The straightforward password solution is sufficient for users with enough discipline to create and remember a long, random password. A token/PIN solution can help to store or generate a long passcode, and this has the additional advantage that it can store or generate different passcodes for multiple applications, and do so on a non-networked, compromise-evident physical device.

A biometric alone is not appropriate for non-network access. Most biometrics have far too little effective keyspace to defend against exhaustive search attacks, and there is no way to authenticate a biometric reader to defend against forgery without a host.

5.3 Authenticating Inside a Security Perimeter

If an authenticator is to be used only within a secure perimeter, some reduced diligence may be suitable. For instance, most portable tokens should be protected by a secret in case they are lost. However, a token mounted in a car for toll payment or garage door entry can be excused from two-factor protection since it is inside a secure perimeter within the car. The same is true for a computer in a house. Since the house has door locks, the computer may not need password protection and the user can choose the option for the machine to "remember" passwords to networked machines. However, there is a danger here. The user must remember to lock her car and house. She must also distinguish between the desktop computer in her house and a portable computer that is only sometimes in her house. Leaving humans to distinguish when and when not to apply different levels of security is dangerous.

5.4 Authenticating Physical Entry with Non-repudiation

Access to a physical location such as a military site or restricted airport area may require stronger assurance that the person possessing the authenticator is its true owner. Since passwords and tokens can be lent or stolen, this is an application for biometrics. A biometric should be combined with a token to store the identity of the user and to protect against the event of biometric compromise. The token offers protection against theft, copying, and replay, and offers compromise detection that the biometric alone does not. It must be assured that the biometric reader is authentic. This can be done by an authentication protocol with a device, or by ensuring that the device is physically secure (e.g., mounted in a wall in a public place and tamper-evident).

One should still be aware that, although a lost token can be changed, a compromised biometric cannot. If attackers can routinely fabricate a copied biometric, one must assume that a token plus biometric system would not be much more secure than a token system alone. (An exception is for cases with a human gatekeeper, in which case use of a fake biometric might be detected.) One should consider this possible downside before investing in and depending upon a biometric system.

5.5 Authenticating Remote Access by Identification

Identification might be used for authentication, but this is impractical using today's technology with anything more than a small number in the identification database. The numbers show the reason why. Consider a grocery store payment application where it is desired that the customers could pay only by giving their biometric. Let's specify that the

system has a maximum of two million users, each of whom would have their biometric template in the database. Using the best false match rate number from Table A2-1 of one in a million for iris recognition, the system false match rate from equation (3) is,

$$FMR(\text{2 million}) = 1.0 - (1.0 - 10^{-6})^{2{,}000{,}000} = 0.86.$$

A false match rate of 86% is unacceptable because too many people will be billed for groceries they didn't buy. Therefore, we make an engineering tradeoff and restrict the user to use his biometric only at his local store (and if shopping elsewhere he must enter his name or a card to perform verification versus identification). We'll assume a modest one thousand biometric users per store. In this case an iris system would have,

$$FMR(1{,}000) = 1.0 - (1.0 - 10^{-6})^{1{,}000} = 0.0009995.$$

This number of about one in a thousand appears much more acceptable. However, if there is an average of 1,000 uses of the system weekly, about one transaction per week will be billed to the wrong person.

To reduce the false match rate further, we can require the person to put down multiple biometrics instead of one. For this, we specify a fingerprint system where the user puts down two different fingers. For independent samples[6], the false match rate multiplies, so assuming the same recognition rate for each finger, the result is the square of that for a single finger. From Table A2-1, $FMR = 0.0001^2 = 10^{-8}$. This low number gives system false match rate of,

$$FMR(1{,}000) = 1.0 - (1.0 - 10^{-8})^{1{,}000} = 10^{-5}.$$

So, at 1,000 transactions per week, there will only be an erroneously assigned bill once every two years. This is more acceptable, but it comes at a cost. Since the false non-match rate adds, the result for two fingers is 2% + 2% = 4%. This means that, at the rate of one transaction per week, each user will be rejected by the system about twice per year.

This is why biometric authentication systems with anything but small numbers in the database require the user to identify herself by card or name, etc., whereupon biometric verification – not identification – is performed.

5.6 Authenticating Remote Access with Non-repudiation

Consider an application for remote electronic access to health records. For privacy sake, it is essential that only the owner and those authorized should be able to access these records. Authorized users will access by phone or computer, preferably without an extra piece of equipment at the client locations. Furthermore, we want an irrefutable record of who has made access to the records. We choose a biometric to defend against repudiation. However, we are averse to using a stable biometric signal because it can't be changed if compromised, and we are not confident that the system can defend against stealing and forging stable biometrics for its technology lifetime.

[6] The independence assumption is weaker for fingerprints from the same person as for fingerprints from different persons. To achieve better independence, a multiple biometric scheme might include two or more *different* biometrics, such as face and fingerprint, which are likely independent.

One authenticator that will meet these specifications is voice used in alterable biometric signal form and operating in a challenge-response protocol. The following is an expansion upon the protocol of Case 4 in Section 3.4, and is shown in Figure 10. Upon user request to authenticate, the system returns a session-specific random number, r, a session-specific challenge, x (which is a random sequence of numbers, letters, or words), a fixed phrase, p, and an encryption function, $E()$. (The fixed phrase could also be a secret if an extra authentication factor were desired.[7]) In step 3, the user speaks the phrase, resulting in signal $BS_1'(p')$, and speaks the challenge resulting in the response, $BS_2'(x')$. The host recognizes x' from $BS_2'(x')$, and extracts biometric templates, B_1' and B_2' from $BS_1'(p')$ and $BS_2'(x')$ respectively. In the fourth step, the host verifies that the responses match correctly: unspoken random number, $r' = r$, and spoken response, $x' = x$. The host matches biometrics, $B_2' \cong B_1'$, to verify that the same person is speaking the response as is speaking the phrase; then it verifies that the person who has spoken the phrase is the same as the one authorized and stored in the user database, $B_1' \cong B(p,U)$. If all these conditions are met, the user is authenticated.

	Client	**Transmission**	**Host**
1	User, U	$U \rightarrow$	
2		$\leftarrow \{r, p, x, E()\}$	r random num.; p, phrase; x, random seq. challenge; $E()$ functions
3	$B_1', p' \rightarrow BS_1'(p')$ $B_2', x' \rightarrow BS_2'(x')$	$E(r', BS_1'(p'), BS_2'(x')) \rightarrow$	$E^{-1}(r', BS_1'(p'), BS_2'(x')) = \{r', BS_1'(p'), BS_2'(x')\}$, $BS_1'(p') \rightarrow BT_1(p') = f(B_1', p')$, $BS_2'(x') \rightarrow BT_2(x') = f(B_2', x')$. Recognize x' from $f(B_2', x')$, Extract B_2' from $f(B_2', x')$, Extract B_1' from $f(B_1', p')$.
4		$\leftarrow$ yes/no	If $r' = r$, and $x' = x$, and $B_2' \cong B_1'$, and $B_1' \cong B(p,U)$, then yes; else no

Figure 10 Challenge-response protocol involving speaker verification of voiced random number.

There are several advantages to this protocol. Since this is an ID-based authenticator whose security depends not upon secrecy but on difficulty to forge, and since it participates in a challenge-response protocol, it is no problem that attackers can hear or record the signal; client, host, eavesdropping, and Trojan horse attacks are unlikely to be successful. Since the challenge-response speech signal cannot be easily lent or stolen, the replay attack is also difficult and this offers defense against repudiation.

[7] If two-factor authentication is desired, the phrase, p, could be a secret. In this case it would not be sent to the user in step 2, but the user would be requested to say the phrase from memory. Note that it is difficult to protect this secret against an eavesdropping attack because it is vocalized.

This protocol uses both text dependent (for *p*) and text independent (for *x*) speech recognition. It is equivalent to applying two speech processing methods to authentication: verbal information verification [50] to verify that the speech-recognized result is the same as the challenge, $x' = x$, and speaker verification [51] to verify that the voice characteristics of the response are close to the user's true characteristics, $B_1'(p) \cong B(p,U)$. Since both these recognition and verification technologies can have errors, there is a question with regards to user inconvenience caused by false non-matches. In Table A2-1, the *FNMR* for text dependent voice is 2%, about on par with the best of the other systems, but *FNMR* for text independent voice is 7%, which is higher than most.

Another question relates to the forge-resistance of speech used in this protocol. It is sufficient just to look at the imperfect recognition results for voice in Table A2-1 to understand that the extraction of robust features for speaker verification (B_1' or B_2') is difficult. An attacker would need to extract features, then use these to synthesize the random sequence response or record the user saying $BS_2'(x')$, to attack the system. Speech synthesis – especially in real time as would be required for this application – presents another level of difficulty to the attacker. So forgery as an attack of this challenge-response voice protocol is arguably more difficult than for stable biometric signals.

6. Conclusions

Password

A single password is an excellent authenticator. Its secrecy is a good defense against theft. It can have a higher keyspace than most other authenticators, and because of this it defends well against search attacks at the client. High keyspace and hashing protect against host attacks. Its ability to participate in challenge-response protocols protects against replay, eavesdropping, and other attacks in transmission. Furthermore, it's convenient and inexpensive.

The main problem is not with a single password, but with multiple passwords. Humans have difficulty remembering these, so they choose easy-to-guess passwords or they write them down and don't safeguard the paper on which they are written. The password advantages evaporate because humans compromise security for the sake of convenience. A more memorable, but lower-entropy password is susceptible to dictionary search attacks. Writing down the password makes it vulnerable to theft. Not only does the strain on human memory makes multiple passwords inconvenient to the user, but administrative costs are high to reset forgotten passwords. As described in Section 2.6, single sign-on will reduce the password memorization burden, but is unlikely to eliminate it totally.

There are two additional shortcomings of passwords. They do not provide good compromise detection, and they do not offer much defense against repudiation.

Token

A token can provide three major advantages when combined with a password. One is that it can store or generate multiple passwords. This changes the task of remembering multiple, changing passwords to one of remembering only the single password needed to access the token: a single sign-on device. A second advantage is that it provides compromise detection since its absence is observable (loss of a password is not). The

third advantage is that it provides added protection against denial of service attacks. For an account with only a password, an attacker can enter incorrect passwords for that user until the account locks out; whereas if combined with token, the attacker cannot just enter incorrect passwords because he has to steal the token first (presumably a more difficult task and one requiring physical presence).

The two main disadvantages of a token are inconvenience and cost. Equipment cost is higher than a password, but comparable to a biometric that requires a reader.

Because of vulnerability to theft, a single-factor token should only be used in special circumstances, such as behind a first line of defense (within a house or restricted office building). A token plus biometric combination has similar security characteristics to a token plus password. However, this combination is likely to cost more due to two required readers, and it may be less convenient (the inconvenience of false non-matches for a biometric versus the inconvenience of remembering a password is a matter of user-preference). If the user needs only to remember a single password, then the relative simplicity and (arguably) better security of the token and password combination is compelling – unless there is a need for non-repudiation.

Biometric

One advantage of a biometric is that it is less easily lent or stolen than the other authenticators, so it provides a stronger defense against repudiation. Since stable biometric signals can be stolen and copied (either now or with higher probability within the lifetime of an implemented system), a biometric should not be deployed in single-factor mode. Furthermore, since biometrics best operate in verification mode, a good second factor choice is a token that stores the identity of the user. The use of biometrics should not give the adopter a false sense of guaranteed non-repudiation. Stable biometric signals have been forged in the past and will be in the future. So a user may be able to repudiate a transaction by claiming forgery.

Attempting to address the vulnerability to theft and forgery of the stable biometric signal, we examined alterable biometric signals employed in a challenge-response protocol in Section 5.6. There are several advantages to using the protocol described in that section. In contrast to stable biometric signals, this authenticator is resistant to forgery and replay. Furthermore, it has the advantage of providing stronger non-repudiation than for stable biometrics. The potential downside of this scheme is that the recognition rate for speaker verification may not be high enough to provide security without inconveniencing the user by many false non-matches.

Recommendations

1. If it is only one password that you need to remember (congratulations on your uncomplicated lifestyle!) and you don’t need to protect against repudiation, then choose a good, high entropy password, memorize it, and keep it secret. There is no need to encumber yourself with a token or deal with the cost of biometrics.
2. If you need to remember multiple passwords, a single sign-on approach is convenient. One option is a token that stores or generates multiple passcodes in a secure manner and is accessed via a single password. The token must be

secure and available when needed. You also have to perform the administrative tasks (backup, etc.). A single sign-on service is a good option for corporate access or Internet access. The tradeoff of service versus token is that the service handles administration for you, but you have some risk that the service may not be secure and may not maintain the privacy of your authentication information as would a privately maintained token.

3. If you are designing a system where it is critical that the person gaining access is the authorized person, or where security against repudiation is desired, then biometrics is a reasonable choice. This should be combined with a token, such as an ID card with the user's identity.
4. No matter what the authenticator choice, it should be emphasized that this is only one component of a full system. The system is only as good as its weakest defense, and multiple lines of defense are better than one. Authentication technologies will continue to progress, as will attackers' technologies. Understand your vulnerabilities, continually monitor for new threats, and react accordingly.

7. Summary

We categorize authenticators by three types according to how they provide security: knowledge-based, object-based, and ID-based. A knowledge-based authenticator provides security by secrecy, and examples are a combination lock and a password. An object-based authenticator provides security by being closely held, and examples are a metal key and an ATM card. An ID-based authenticator provides security by uniqueness and copy-resistance, and examples include a passport and a biometric.

We compare authenticators with respect to potential attacks and other issues. The attacks include: client and host search attacks, eavesdropping, theft (including biometric forging), replay, Trojan horse, and denial of service. Other security issues include: non-repudiation, compromise detection, and the administrative issues of registration/enrollment, reset or compromise recovery, and revocation.

Although an appropriate authentication solution depends upon the particular application, a few combinations of authenticators are recommended. One is the simple password, which has very high security – if the user can remember it. Another is the token and password combination, especially if the token can store or generate multiple passwords and act as a personal single sign-on device. A third is a biometric in combination with a token if non-repudiation is required, and an alterable biometric signal used in a challenge-response protocol is recommended for the biometric in this case.

Acknowledgements:

My thanks to those who have made helpful comments on this work or drafts of this paper: Jim Esch, Fred Juang, Colin Mallows, Mahalingam Mani, L. J. O'Gorman, Jonathon Phillips, Mohan Sondhi, and Jim Wayman.

References:

1. D. Kahn, The Codebreakers: The Story of Secret Writing, Scribner, New York, 1996.
2. G. Stocksdale, "NSA glossary of terms used in security and intrusion detection," SANS Institute Resources, http://www.sans.org/newlook/resources/glossary.htm
3. E. Rescorla, "SSL and TLS: Designing and Building Secure Systems," Addison-Wesley, Massachusetts, 2000.
4. Federal Information Processing Standards Publication, FIPS-197, "Specification for the Advanced Encryption Standard", NIST, Nov, 2001. http://csrc.nist.gov/encryption/aes/
5. R. Morris, K. Thompson, "Password security: A case history," *Comm. ACM*, Vol. 22, no. 11, Nov. 1979, pp. 594-597.
6. B. L. Riddle, M. S. Miron, J. A. Semo, "Passwords in use in a university timesharing environment," *Computers and Security*, Vol. 8, no. 7, 1989, pp. 569-579.
7. D. L. Jobusch, A. E. Oldehoeft, "A survey of password mechanisms: Weaknesses and potential improvements," *Computers and Security*, Vol. 8, no. 8, 1989, pp. 675-689.
8. 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.
9. M. Bishop, D. V. Klein, "Improving system security via proactive password checking," *Computers and Security*, Vol. 14, no. 3, 1995, pp. 233-249.
10. J. Bunnell, J. Podd, R. Henderson, R. Napier, J. Kennedy-Moffat, "Cognitive, associative, and conventional passwords: Recall and guessing rates," *Computers and Security*, Vol. 16, no. 7, 1997, pp. 645-657.
11. 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.
12. R. Pond, J. Podd, J. Bunnell, R. Henderson, "Word association computer passwords: The effect of formulation techniques on recall and guessing rates," *Computers and Security*, Vol. 19, no. 7, 2000, pp. 645-656.
13. R. E. Smith, Authentication, From Passwords to Public Keys, Addison-Wesley, Boston, 2002, pp. 255-284.
14. A. Jain, R. Bolle, S. Pankanti (ed.s), Biometrics: Personal Identification in Networked Society, Kluwer Press, The Netherlands, Nov. 1998.
15. S. Pankanti, R. M. Bolle, A. Jain, "Biometrics: The future of identification," special issue of *Computer*, Vol. 33, no. 2, Feb. 2000.
16. R. M. Bolle, J. H. Connell, S. Pankanti, N. K. Ratha and A. W. Senior, Guide to Biometrics: Selection and System Design, Springer-Verlag, New York, 2003.
17. R. Anderson, Security Engineering, Wiley Computer Publishing, New York, 2001, pp. 384, 398-399.
18. W. Stallings, Cryptography and Network Security: Principles and Practice, 2nd Edition, Prentice Hall, New Jersey, 1999, p. 490.
19. B. Schneier, Applied Cryptography, Wiley Publ., New York, 1996, pp. 429-459.

20. K. P. Weiss, “Method and apparatus for positively identifying an individual,” U.S. Patent 4720860, 19 Jan. 1988.
21. L. O’Gorman, “Seven issues with human authentication technologies,” *IEEE Workshop on Automatic Identification Advanced Technologies*, Tarrytown, New York, March 2002, pp. 185-186.
22. T. Matsumoto, H. Matsumoto, K. Yamada, S. Hoshino, "Impact of Artificial Gummy Fingers on Fingerprint Systems," *Proceedings of SPIE*, Vol. 4677, San Jose, Feb. 2002, pp. 275-289.
23. N. K. Ratha, J. H. Connell, and R. M. Bolle, “Enhancing security and privacy in biometrics-based authentication systems,” *IBM Systems J*., Vol. 40, No. 3, 2001.
24. S. M. Bellovin, M. Merritt, “Encrypted key exchange: Password-based protocols secure against dictionary attacks,” *Proc. 1992 IEEE Computer Society Conference on Research in Security and Privacy*, 1992, pp. 72-84.
25. L. Gong, M. A. Lomas, R. M. Needham, J. H. Saltzer, “Protecting poorly chosen secrets from guessing attacks,” *IEEE Journal on Selected Areas in Communications*, Vol. 11, no. 5, Jun., 1993, pp. 648-656.
26. Thomas Wu. “The secure remote password protocol,” *Proc. of the 1998 Internet Society Network and Distributed System Security Symposium*, pp. 97-111, San Diego, CA, March 1998. http://citeseer.nj.nec.com/article/wu98secure.html
27. S. Pankanti, S. Prabhakar, and A. K. Jain, “On the Individuality of Fingerprints”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2002 (in Press).
28. CCIMB, Common Criteria Interpretations Management Board, “Common Criteria for Information Technology Security Evaluation, Part 2, Security Functional Requirements”, CCIMB-99-032, Aug. 1999, p. 8. http://www.commoncriteria.org/docs/PDF/CCPART2V21.PDF
29. S. M. Matyas Jr., J. Stapleton, “A biometric standard for information management and security,” *Computers and Security*, Vol. 19, no. 5, 2000, pp. 428-441.
30. B. Schneier, “Biometrics: truths and fictions,” *Crypto-gram*, 15-Aug-98 http://www.counterpane.com/crypto-gram-9808.html
31. R. Germain, A. Califano, S. Colville, “Fingerprint matching using transformation parameter clustering,” *IEEE Computational Science and Engineering*, Vol. 4, No. 4, 1997, pp. 42-49.
32. J. L. Wayman, “Error-rate equations for the general biometric system,” *IEEE Robotics and Automation Magazine*, Vol. 6, No. 1, March 1999, pp. 35-48.
33. C. Rigney, S. Willens, A. C. Rubens, and W. A. Simpson, “Remote Authentication Dial In User Service (RADIUS),” RFC 2865, Internet Engineering Task Force, June, 2000, http://www.ietf.org/rfc/rfc2865.txt?number=2865
34. U. D. Black, Internet Security Protocols: Protecting IP Traffic, Prentice Hall, 2000, New Jersey, pp. 113-121.
35. Rivest, R. and S. Dusse, "The MD5 Message-Digest Algorithm", RFC 1321, Internet Engineering Task Force, April 1992, http://www.ietf.org/rfc/rfc1321.txt?number=1321
36. J. G. Steiner, B. Clifford Neuman, and J.I. Schiller, “Kerberos: An Authentication Service for Open Network Systems,” *Proc. Usenix* , February, 1988. (Version 4).

37. J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, Internet Engineering Task Force, September, 1993, http://www.ietf.org/rfc/rfc1321.txt?number=1321
38. R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, Vol. 21(12), December, 1978, pp. 993-999.
39. J. Taschek, "Liberty Alliance or Passport?" *eWeek*, 24 June, 2002, http://www.eweek.com/article2/0,3959,266840,00.asp
40. P. Hallam-Baker, E. Maler (ed.s), "Assertions and protocol for the OASIS security assertion markup language (SAML)," Committee Specification 01, cs-sstc-core-01, 31 May 2002, http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf
41. C. E. Shannon, "A mathematical theory of communication. *Bell System Tech. J.*, Vol. 27, July, October, 1948, pp. 379-423, 623-656, http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html.
42. D. R. Stinson, Cryptography Theory and Practice, 2nd edition, Chapman & Hall/ CRC Press, 2002.
43. National Institutes of Science and Technology, "Secure Hash Standard", NIST FIPS Publication 180-1, US Department of Commerce, April 1995.
44. R. M. McCabe, "ANSI/NIST-ITL 1-2000 Data Format for the Interchange of Fingerprint, Facial, and Scar Mark & Tattoo (SMT) Information," ftp://sequoyah.nist.gov/pub/nist_internal_reports/sp500-245-a16.pdf
45. R. Shirey, "Internet Security Glossary," RFC 2828, Internet Engineering Task Force, May 2000.
46. Microsoft Knowledge Base Article – 102716, "User authentication with Windows NT," Aug. 2001. http://support.microsoft.com/default.aspx?scid=kb%3ben-us%3b102716
47. M. Howard, Designing Secure Web-Based Applications for Microsoft Windows 2000, Microsoft Press, 2000, pp. 407-421.
48. Oxford English Dictionary, Second Edition, (with Vol. 1-3 Additions), Oxford University Press, 2002.
49. R. Derakhshani, S. Schuckers, L. Hornak, L. O'Gorman, "Determination of Vitality from A Non-Invasive Biomedical Measurement for Use in Fingerprint Scanners," *Pattern Recognition*, Vol. 36, No. 2, Feb. 2003, pp. 383-396.
50. Q. Li, B-H. Juang, Q. Zhou, and C.-H. Lee,, "Automatic verbal information verification for user authentication,", *IEEE Transactions on Speech and Audio Processing*, Vol. 8, no. 5, Sept, 2000, pp. 585-596.
51. L. Rabiner, B-H. Juang, Fundamentals of Speech Recognition, New Jersey, USA, Prentice Hall, 1993.
52. R. L. van Renesse, Optical Document Security, Artech House, Massachusetts, 1994.
53. A. Martin, M. Przybocki, "The NIST 1999 Speaker Recognition Evaluation-An Overview," *Digital Signal Processing*, Volume 10, Numbers 1-3, 2000, pp. 1-18, http://www.idealibray.com/links/toc/dspr/10/1/0
54. D.M. Blackburn, J.M. Bone, and P.J. Phillips, "FRVT 2000 Evaluation Report," Feb. 2001, www.dodcounterdrug.com/facialrecognition/DLs/FRVT_2000.pdf

55. D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, A.K. Jain, "FVC2000: Fingerprint Verification Competition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, to be published, 2001. http://bias.csr.unibo.it/fvc2000/Downloads/fvc2000_report.pdf
56. T. Mansfield, G. Kelly, D. Chandler, J. Kane, "Biometric product testing final report," *CESG report*, March, 2001, www.cesg.gov.uk/technology/biometrics

Appendix 1 – The Paradox of Secure Biometrics

The static nature of stable biometric signals suggests "the paradox of secure biometrics:"

1. A person's biometric is stable and distinctive, and these qualities make it a good authenticator.
2. However, stability leaves no option for compromise recovery, since you cannot change a biometric if stolen.
3. Furthermore, since a biometric is not secret, its information can be learned and copied; and worse yet, since it is distinctive, the biometric alone gives information on who to attack.
4. So, are stability and distinctiveness really desirable characteristics of a good authenticator?

In point 1, stability refers to the fact that a good biometric maintains its distinctive features over time. For instance, fingerprint and iris features are formed in the womb and do not change throughout life. Face and voice features are stable through most of mature life. Note that we use the term "stable" rather than the more often claimed "immutable" for biometrics. Though good biometric features do not change throughout life (at least mature life), this does not mean that it is immutable since acid or plastic surgery can alter a biometric. We use the term "distinctive" rather than the more often claimed "unique" for biometrics. Although no evidence exists of two different fingerprints ever matching, non-zero false match rates for all biometric algorithms to date show that biometrics are not unique to the resolution of current computer methods, which is what concerns us here.

Point 2 states that compromise recovery is not possible for a stable biometric. Compromise recovery is analogous to intrusion detection, because both assume that no matter how strong the security design, successful attacks will occur and a good design should be prepared for this [17, 18]. With this expectation, recovery plans can be made in case a security layer is compromised. When a credit card is lost, for instance, it is canceled as soon as possible and a new card issued with a different number. However, one cannot reissue a stable biometric.

In point 3, the combination of lack of secrecy [28] and distinctiveness also presents a problem for biometrics. Consider this analogy. If you lose a slip of paper upon which you have written a PIN, you might be only mildly concerned since the finder likely won't know to which account it is associated. If you lose a slip of paper with a PIN and your account ID, you will be more concerned. Now, if you tattoo that PIN and account number

onto your forehead, an attacker can capture this information just by looking at your face. Most stable biometrics (at least face, iris, and fingerprint) are like this tattoo.

To present both sides, there are forgery detection methods that reduce the ability to use stolen biometric features [49]. To date, many of these anti-forgery methods have been defeated [22], however it is shortsighted to argue which side is currently winning, counterfeiters or anti-counterfeiters. For currency protection, anti-counterfeiting is a perpetual cycle: authorities design good anti-counterfeiting protections, then attackers devise counterfeiting schemes around these protections, then authorities devise stronger protections, etc. [52] The difference with biometrics is that we cannot change our body features to improve their counterfeit resistance.

Appendix 2 – Biometric Error Rates

We include biometric error rate statistics in this section to help with authenticator comparisons when biometrics are involved. Statistics are derived from four test studies performed by respected, 3rd-party sources. These are listed below by: name; main sponsoring organization; date of testing; biometric type; some test descriptors; test population size; and reference.

- **NIST Speaker '99**; NIST; Mar.-Apr. 1999; voice; telephone quality, variable channel/handset quality, text independent, up to 1 minute duration; 233 target trial speakers, 529 imposter trial speakers (test "1-Speaker Detection"); [53].
- **FRVT 2000** (Facial Recognition Vendor Test); DARPA; Mar.-Jun. 2000; face; mugshot pose, ambient probe lighting, mugshot gallery lighting, time separation 11-13 months (test "T3"); 467 probe faces, 227 gallery faces; [54].
- **FVC 2000** (Fingerprint Verification Competition); University of Bologna; Jun.-Aug 2000; fingerprint; 500dpi, 256x364 size capacitive sensor (test "DB2"); 100 fingerprints; [55].
- **CESG Biometric Testing Report**; CESG; May-Dec. 2000; face, fingerprint, hand, iris, vein, and voice; standard verification mode of operation for each system, failure-to-enroll removed, time separation 1-2 months; about 200 subjects; [56].

We have extracted some results from these tests, shown in Table A2-1, to be used in examples of Section 5. A few caveats must be given with respect to the selection process. With so many variations in test design, population characteristics, etc., it is impossible to choose the single, "right" data. We chose operating points described by (*FMR, FNMR*) error rate pairs that apply to some practical situations; for example, a single-attempt *FNMR* in the range of 1-3% is reasonable for many applications. Where the error rates were higher, as for face and NIST voice, we chose the equal error rate. At that chosen operating point, we chose the best results of all products (the best product at that operating point). However, we were not so generous in making the choice from different testing variables. We chose the most challenging, but practical, variable. For example, for face verification in FRVT, we chose a temporal test (verification separated from enrollment by about a year), whose results showed much more challenge to the different

systems than for other testing variables. The chosen results from these tests enable us to show examples of expected performance under similar conditions in the examples of Section 5. But different applications, different devices, product improvements, etc. may give better or worse performance.

Another caveat must be made with respect to Table A2-1. One cannot compare the results of different biometrics outside of the bounds of a single test (because of differences in test design, subjects, etc.). It is evident in the table that results vary widely when different tests were made on the same biometric type. Where two different tests are run on a single biometric, we have attempted to identify a test feature that contributes to this difference in the column titled, "Test Parameter,"

One might notice that the CESG results are uniformly better than each other test on the same biometric type. We suggest two reasons for this. One is the use of different testing parameters as noted. For instance, for the speaker verification tests, we would expect the NIST results with text independence and channel/handset variability to be worse than the CESG results where the text was known and equipment the same. The other reason is that for the CESG testing, data was collected with the same system for which matching was performed. Conversely, for the FRVT, FVC, and NIST tests, data collection was separate from matching, so there wasn't the ability of a particular product to be tuned to the same data it collected.

Finally, the CESG results for iris actually yielded 0% FMR for the 200 subjects tested. Because of this, the CESG authors use the manufacturer's claimed results that were based on a larger sample size.

Table A2-1 Recognition error rate pairs chosen from results of benchmark testing for several biometrics and from different tests.

Biometric	Test	Test Parameter	Attempts	FNMR	FMR
Face	FRVT [54]	11-13 mo. spaced	1	16%	16%
	CESG [56]	1-3 mo. spaced	3	6%	6%
Fingerprint	FVC [55]	Mainly age 20-30	1	2%	0.02%
	CESG [56]	Mainly age 30+	3	2%	0.01%
Hand	CESG [56]	-	1	3%	0.3%
	CESG [56]	-	3	1%	0.15%
Iris	CESG [56]	-	1	2%	0.0001%
	CESG [56]	-	3	0.25%	0.0001%
Voice	NIST [53]	Text independent	1	7%	7%
	CESG [56]	Text dependent	3	2%	0.03%