A Fuzzy Commitment Scheme

Ari Juels
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730
E-mail: `ari@rsa.com`

Martin Wattenberg
328 West 19th Street
Apt. 2C
New York, New York 10011
E-mail: `w@bewitched.com`

September 25, 2013

Abstract

We combine well-known techniques from the areas of error-correcting codes and cryptography to achieve a new type of cryptographic primitive that we refer to as a *fuzzy commitment* scheme. Like a conventional cryptographic commitment scheme, our fuzzy commitment scheme is both *concealing* and *binding*: it is infeasible for an attacker to learn the committed value, and also for the committer to decommit a value in more than one way. In a conventional scheme, a commitment must be opened using a unique *witness*, which acts, essentially, as a decryption key. By contrast, our scheme is *fuzzy* in the sense that it accepts a witness that is close to the original encrypting witness in a suitable metric, but not necessarily identical.

This characteristic of our fuzzy commitment scheme makes it useful for applications such as biometric authentication systems, in which data is subject to random noise. Because the scheme is tolerant of error, it is capable of protecting biometric data just as conventional cryptographic techniques, like hash functions, are used to protect alphanumeric passwords. This addresses a major outstanding problem in the theory of biometric authentication. We prove the security characteristics of our fuzzy commitment scheme relative to the properties of an underlying cryptographic hash function.

1 Introduction

Cryptographic protocols are conventionally predicated on exact knowledge. An authentication system using RSA signatures, for example, derives its security largely from the presumption that a legitimate user with public key (N, e) possesses a corresponding secret key of the uniquely specifiable form (N, d). There are situations, however, in which human and other factors undermine the possibility of exactness in a security system. In biometric systems where users identify themselves by means of fingerprint features, for example, variability in user interaction is such that a finger is rarely read exactly the same way twice. Moreover, even if knowledge in a system is exact, its transmission may only be approximate. Users typically make typing errors, for example, when entering passwords on keyboards. Similarly, data transmission channels are often subject to random noise. Our aim in this paper is to describe a simple cryptographic primitive, namely a type of commitment scheme, that uses well-known algorithms to facilitate the use of approximate information in cryptographic systems. As a model for approximate reasoning in humans, researchers in artifical intelligence have elaborated a notion known as "fuzzy logic" [37]. By analogy, we call the primitive introduced in this paper a *fuzzy commitment* scheme.

In a conventional *bit commitment* scheme, one player, whom we denote the *sender*, aims to conceal a bit b. The sender produces an encryption of b, denoted by y, and sends y to a second player, known as the *receiver*. A bit commitment scheme should be such that it is infeasible for the second player to learn the bit b. Additionally, the sender should later be able to "open" the commitment y, that is, to prove to the receiver that y indeed represents an encryption of b. It should only be feasible, however, for the sender to "open" y in one way, that is, to decrypt a unique value of b. We may view this, intuitively, as a process whereby the sender places the bit b in a safe and gives the safe to the receiver. Only the sender can open the safe, since she alone knows the combination. Moreover, she cannot change the value contained in the safe while it is in the keeping of the receiver.

Formally, a bit commitment scheme consists of a function $F : \{0,1\} \times X \rightarrow Y$. To commit a bit b, the sender chooses a *witness* $x \in X$, generally uniformly at random. The sender then computes $y = F(b, x)$. This value y is known as a *blob*. It represents the bit b sealed in a "safe". To "open" or *decommit* the blob y, the sender produces the bit b and the witness x. The blob is successfully opened if the receiver has been convinced that y indeed represents an encryption of b.

A bit commitment scheme is said to be *concealing* if it is infeasible for the receiver to guess b with probability significantly greater than $1/2$. It is said to be *binding* if it is infeasible for the sender to decommit the blob y with the incorrect bit, that is, with $1 - b$. Note that it is possible to deploy a bit commitment scheme as a commitment scheme on an arbitrarily long string of bits by committing each bit independently. We shall use the term *commitment scheme* in this paper to refer to a scheme that involves commitment of a bit string c (or other potentially non-binary value) in a single blob, and for which it is possible to extract c efficiently given a witness for the blob. Thus we assume $F : C \times X \rightarrow Y$, where B is some potentially non-binary space. Additionally, our scheme will be such that production of a valid witness x' allows the committed value c to be efficiently determined from a commitment $F(c, x)$. This is not the case in general for commitment schemes; often, both c and a valid witness

x' are required to enable the sender to prove that that $F(c, x)$ represents a commitment of c. Finally, we offer a stronger notion of binding than that conventionally employed in the literature. We require not just the infeasibility of decommitting two distinct values c and c' from a single commitment, but also that decommitment using two substantially different witnesses be infeasible. This is discussed in detail in Section 5. For further details on bit commitment, the reader may consult a standard cryptography textbook, such as [33], or one of the seminal papers on the subject, such as [12].

Our aim in designing a *fuzzy* commitment scheme F is to achieve a new property that we loosely call "fuzziness". By this, we mean that the commitment scheme should be resilient to small corruptions in witness values. More precisely, we aim to allow a blob $y = F(b, x)$ to be opened using any witness x' that is close to x in some appropriate metric, such as Hamming distance, but not necessarily identical to x. At first glance, the requirement for this type of resilience seems contradictory to the requirements that F be binding and concealing. After all, to achieve these two security aims, F must be an encryption function of sorts. It would therefore seem necessary, in accordance conventional encryption or hash function design, for small changes in input values to yield large, unpredictable changes in output values. In other words, F should thoroughly and unpredictably "scramble" input bits. On the other hand, the requirement of fuzziness in F suggests exactly the opposite, namely a high degree of local structure. In this paper, we show how to reconcile these ostensibly conflicting goals using well-known components drawn from error-correcting codes and cryptography. We combine a conventional hash function h with an error-correcting code used in a somewhat unorthodox way. Our construction is quite simple, and provably secure with respect to the underlying hash function h.

1.1 Organization of this paper

In Section 2, we give an overview of biometric authentication and a description of related work. We provide a brief introduction to error-correcting codes in Section 3. We describe our fuzzy commitment construction in Section 4, and also discuss some applications to general security protocols. In Section 5, we state theorems regarding the security characteristics of our construction and analyze its resilience. We conclude in Section 6 with some suggestions for future areas of research. Short proofs of our theorems are provided in the appendix.

2 Background

2.1 Biometrics

An important motivation for our investigation of fuzzy commitment is the problem of secure storage of data in biometric systems. We now give a brief overview of this area.

Biometric authentication is the process of establishing the identity of an individual using measurements of some collection of his or her biological characteristics. Applied in its broadest sense, biometric authentication describes the processes that human beings use naturally to recognize one another, primarily through the senses of sight and hearing. When you recognize a friend by her face, you are performing a type of biometric authentication.

Biometric authentication can also assume automated forms involving the identification of individuals to computer systems by such means as retinal and fingerprint scans. Until recently, biometric technologies have been the preserve of government agencies and science fiction movies, as in [2, 10, 27]. Recent improvements in on-chip scanning technologies as well as a proliferation of peripheral devices such as microphones and video cameras in desktop systems computers have promised to bring automated biometric authentication technologies to a consumer level in the near future [11, 19]. A plethora of relatively inexpensive biometric authentication techologies are now available, including ones based on fingerprint scanning, iris scanning, voice authentication, face recognition – and even body odor. These technologies promise to play a major role in a broad range of data security applications.

Much of the appeal of biometric authentication is its promise of heightened security relative to passwords. As security specialists know well, users often choose passwords poorly and write them down in conspicuous places, making them vulnerable to attack. Biometrics eliminate the problem of forgotten passwords and, according to industry claims, are largely resistant to remote capture.

Biometrics, however, pose a security risk that passwords do not. In many operating systems, as in most implementations of UNIX, a given password P is not stored explicitly in the system password file. Instead, a commitment of P is stored in the form of a hash $h(P)$ [18, 26].[1] (Note that this hash may be regarded as a commitment on a null value for which P is the witness.) Thus it is possible to verify that a user has entered her password correctly, while even a system administrator cannot feasibly extract a well-chosen password P from the password file entry $h(P)$. Protecting user secrets through a straightforward means of commitment like hashing is not possible, though, for biometric authentication. The reason is this: two readings of the same biometric are rarely identical. Changes occur naturally in biological characteristics over time. Additionally, there is substantial variability in human execution of physical tasks.

[1]Hashed passwords are typically also salted as a defensive measure against dictionary attacks.

Because users are inconsistent in the position and pressure with which they apply their fingers to readers, for example, fingerprint reading devices almost always extract different information from multiple readings of the same finger – even when these readings occur in rapid succession.

To handle the variability inherent in biometric authentication, most systems store for each user what is called a *template.* The template x_U for user U consists of a biometric reading or set of readings obtained from U during an initial registration or *enrollment* process. When a user claiming to be U later authenticates herself, resulting in biometric reading x', a matching algorithm is invoked to compare x' with x_U and determine whether the two belong to the same user. How much x' must look like x_U to generate a match depends on the matching algorithm and its parameterization. The parameterization of a matching algorithm depends in turn on the false rejection and false acceptance rates desired in a given authentication system.

Because of the resilience required for biometric authentication systems, templates are usually stored, unlike passwords, in explicit form. Yet the protection of biometric information is far more critical than that of passwords. It is easy to use separate passwords for different systems, and to change passwords on a frequent basis. Using multiple biometrics across systems and changing biometric passwords is harder. In a system employing fingerprints, for example, a user can change her "password" at most nine times. Additionally, many users have serious concerns about the threat to privacy posed by compromised biometric information. These issues have been persistent points of contention in the development of biometric authentication systems [11, 19].

2.2 Related work

The idea of fuzziness in commitment schemes perhaps first arises in the literature in connection with "collisionful" hash functions, intended for use in password protection. (Recall that the hash of a password may be viewed as a commitment.) "Collisionful" hash functions, introduced in [9], aim to discourage guessing attacks against passwords by means of a dense pre-image space. Gong [20] describes methods of carefully determining collision sets for this purpose, enabling the selection of multiple, plausible passwords (or witnesses) as pre-images for a given hash value. Other research in this area includes that of Bakhtiari *et al.* [3, 4, 5].

As mentioned above, error-correcting codes play a central role in our fuzzy commitment construction. The application of error-correcting codes to cryptography has a long history. Error-correcting codes are particularly important in non-standard cryptographic models. They serve, for example, as a means of eliminating errors introduced by "dark counts" and other apparatus faults in quantum cryptographic key distribution protocols (see, e.g., [6]). They are likewise a critical component in the implementation of oblivious transfer and key agreement protocols over both quantum [7, 14] and noisy channels (see, e.g.,

[13]).

Error-correcting codes can also be employed in the construction of traditional cryptographic primitives. In [24], McEliece elaborates a well-known public-key cryptosystem whose hardness is based on the NP-hard problem of decoding an arbitrary linear code [8]. Researchers have also proposed identification [32] and digital signature schemes [1] based on error-correcting codes, among other applications. In a recent paper [21], Jakobsen demonstrates that a class of error-correcting codes known as Reed-Solomon codes can even assist in the cryptanalysis of block ciphers.

A notable feature of these efforts is their use of error-correcting codes to subserve conventional cryptographic goals. In an important divergence from this tradition, Davida, Frankel, and Matt [16] propose a synthesis of error-correcting codes with cryptographic techniques to achieve a new and somewhat unusual security goal. They describe a system in which a biometric template can be stored in non-explicit, protected form, but such that some corruption in subsequent readings can be tolerated. They achieve this by computing check bits on the template using a linear error-correcting code, and storing these check bits along with a hash of the template. Their construction offers important new ideas, and may in fact be regarded as a kind of fuzzy commitment. Their system does not have the necessary error tolerance to work in many real-world applications. They require, for instance, that a biometric scan be repeated many times in succession under the assumption that the errors in these scans will be wholly independent. Follow-up analysis of their work may be found in [17].

Vendors of biometric systems have for some time recognized the importance of achieving a practical system along the lines of that proposed by Davida *et al.* To this end, the company Mytec Technologies has developed a related technology, consisting of an encryption process in which biometric data serves as an unlocking key. Sold under the brand name Bioscrypt™, this technology overcomes the problem biometric data corruption by means of Fourier transforms. While fairly efficient, however, it carries no rigorous security guarantees (see, e.g., [30, 31]).

Our work on fuzzy commitment may be regarded as an improvement on and generalization of that of Davida *et al.* As mentioned above, their scheme involves the extension of a biometric template into an error-correcting codeword through the addition of check bits. (See Section 3 for the definition of a codeword.) In contrast, our fuzzy commitment scheme, as applied to biometric templates, treats the template itself without any modification as a corrupted codeword. This difference in perspective yields several advantages. Most importantly, our construction links the number of codewords to the security parameter, while that of Davida *et al.* links it to the significantly larger message (i.e., template) size. In consequence, our construction uses much smaller error-correcting codes than that of Davida *et al.* and achieves significantly higher resilience. Our fuzzy commitment construction thereby brings the idea of secure biometric template storage farther into the realm of practical application.

3 Error-Correcting Codes

To provide background for the fuzzy commmitment construction presented in the next section, we now give a brief overview of *error-correcting codes*. The goal of an error-correcting code is to enable transmission of a message m intact over a noisy communication channel. This is accomplished by mapping m to a longer string c prior to transmission. The string c is constructed so as to contain redundant elements. Therefore, even if some of the bits of this string are corrupted by noise, it remains possible for a receiver to reconstruct c, and consequently the message m.

More formally, an error-correcting code consists of a set $C \subseteq \{0,1\}^n$ of *codewords*. This set contains the strings to which messages are mapped prior to transmission. Hence, in a code for use with k-bit messages, C contains 2^k distinct elements. To achieve redundancy, it is a requirement that $n > k$. Error-correcting codes may of course be easily defined on non-binary spaces as well, and our constructions are straightforwardly extensible to such spaces.

To use an error-correcting code, we require functions for encoding and decoding of messages. Let $M = \{0,1\}^k$ represent the space of messages. The function $g : M \rightarrow C$, which we call a *translation function*, represents a one-to-one mapping of messages to codewords. In other words, g is the mapping used prior to message transmission. (Conversely, g^{-1} is used upon message receipt to retrieve the transmitted message from a reconstructed codeword.) The function $f : \{0,1\}^n \rightarrow C \bigcup \{\phi\}$, known as a *decoding* function, is used to map arbitrary n-bit strings to codewords. When successful, f maps a given n-bit string x to the nearest codeword in C (i.e., nearest in terms of Hamming distance).[2] Otherwise, f fails, and outputs ϕ.[3]

The robustness of an error-correcting code depends upon the minimum distance between codewords. To make this idea precise, we require some basic notation regarding strings of binary digits. Let the symbol $+$ (and equivalently, the symbol $-$) denote the bitwise XOR operator on bitstrings. (In this context, the symbols $+$ and $-$ are more intuitively appealing than $\oplus$.) The *Hamming weight* of an n-bit string u, denoted by $\| u \|$, is defined to be the number of '1' bits in u. The *Hamming distance* between two bitstrings u and v is likewise defined to be the number of digits in which the two strings differ. Equivalently, the Hamming distance is equal to $\| u - v \|$.

We say that a decoding function f has a *correction threshold* of size t if it can correct any set of up to t bit errors. More precisely, for any codeword

[2]The task of mapping an arbitrary string to its nearest codeword is known as the *maximum likelihood decoding* problem. Practical classes of codes with polynomial-time solutions to this broad problem are at present unknown. Conventional decoding functions perform a more limited task: they successfully decode any word that lies within a certain radius of some codeword. This is all that our fuzzy commmitment algorithm requires.

[3]Error correcting codes may work somewhat differently. For example, with use of *list decoding*, f may yield a set of candidate codewords, rather than a single correct one. The underlying principles in our construction remain the same in such settings.

$c \in C$ and any error or *offset* $\delta \in \{0,1\}^n$ with $\| \delta \| \leq t$, it is the case that $f(c+\delta) = c$. We say that a code C has a correction threshold of size t if there exists a decoding function f for C that has correction threshold t. Observe that the distance between any two codewords in C must be at least $2t+1$. We define the *neighborhood* of a codeword c to be $f^{-1}(c)$. In other words, the neighborhood of c consists of a subset of the n-bit strings that f maps to c. The decoding function f is generally such that any codeword in $f^{-1}(c)$ is closer to c than to any other codeword.

Example 1 *Let* $n = 3, k = 1$*, and* $C = \{000, 111\}$. *Let the decoding function* f *compute majority, i.e.,* f *maps a bitstring* $x \in \{0,1\}^3$ *to* 000 *if at least two digits of* x *are* 0 *and to* 1 *if at least two are* 111. *This decoding function has* $t = 1$. *In other words,* f *can correct a single error, since changing a single digit in either* 000 *or* 111 *does not change the majority.* ∎

The ratio k/n in an error-correcting code is known as its *coding efficiency*, and measures the degree of redundancy in the code. (The lower the coding efficiency, the more redundancy in the codewords.) The $\{000, 111\}$ code, for instance, has a coding efficiency of $1/3$. In general, codes that can correct a large number of errors must have a low coding efficiency.

Further details on error-correcting codes are available in any of a number of textbooks on the topic, such as, e.g., [23, 28, 36].

3.1 How we use error-correcting codes

As explained above, an error-correcting code traditionally involves changing a message to a codeword before transmitting it across a noisy channel. In some situations, however, this initial encoding step is impossible because the message cannot be modified. For instance, in the case of biometric identification the noisy channel might be an error-prone fingerprint reading machine, and the "message" might be an actual fingertip. Thus, we do not have the ability to add redundancy to the "message". Because this constraint arises in our use of fuzzy commitment, we treat a witness (e.g., biometric template) as a corrupted codeword, rather than a message. In consequence, our construction does not map messages from the space M to the set of codewords. In fact, we do not make use of M at all. Rather, we make use of only half of an error-correcting code: we use the decoding function f, but not really the translation function g. This use of error-correcting codes is somewhat unorthodox. It represents the novel element in our construction.

The commonest class of error-correcting codes consists of what are known as *linear codes*. These are codes whose set of codewords, in the binary case, forms a vector space over the field with two elements. Almost all of the error-correcting codes used in practice are linear. Although not strictly necessary, it is for several reasons convenient to choose a linear code for our construction.

For example, one property of linear codes useful in a number of applications of our fuzzy commitment construction, as we shall see, is that it is very easy to select a codeword c uniformly at random from C.

4 Construction of our fuzzy commitment scheme

4.1 Intuition

Let us now describe the construction of our fuzzy commitment scheme F. We shall construct F so as to commit a codeword c using a witness x, where both c and x are n-bit strings.

Observe that an n-bit witness x can be uniquely expressed in terms of the codeword (committed value) c along with an offset $\delta \in \{0,1\}^n$ such that $x = c + \delta$. Given a witness x expressed in this way, the idea behind the fuzzy commitment function F is to conceal c using a conventional hash function h, while leaving δ in the clear. The information δ provides resilence in the witness required to open F. In particular, δ provides some partial information about x. On the other hand, the remaining information needed to specify x, namely the codeword c, is presented in a concealed form as $h(c)$.

Recall that we define $|C| = 2^k$. The amount of information contained in the codeword c, and thus the amount of information about the witness x concealed in $h(c)$ depends on k, that is, on the number of codewords in C. The greater the number of codewords, the greater the amount of information about the witness x that is concealed in $h(c)$. In contrast, the amount of information in δ determines the level of resilience in F. If we are presented with a witness x' that is near x, we can use δ to translate x' in the direction of x, facilitating our recovery of the committed codeword c. As we shall see, we achieve a tradeoff between resilience and security by varying k, and thus the relative distribution of information between δ and c.

In biometric scenarios, x will typically represent a biometric template, such as a fingerprint. The codeword c will represent a secret key protected under this template. For example, c might be a decryption key protected under the user's fingerprint x as the commitment $F(c,x)$. In order to unlock and reveal this key, it suffices for the user to present a corrupted fingerprint image x' sufficiently close to x. Note that in some scenarios where is not necessary to protect c itself, the codeword c must still be drawn from a large space C, in order to conceal the witness x. Consider, for example, a straightforward fingerprint authentication scenario meant to model the use of hashed passwords on UNIX systems. Here, $F(c,x)$ is stored on a server. In order to demonstrate her identity, it suffices for the user simply to present to the server a fingerprint image that succesfully decommits $F(c,x)$. The committed value c does not serve here as a cryptographic key. Nonetheless, c must be drawn from a large enough space C to ensure that $F(c,x)$ does not reveal x. If $|C|$ (or, equivalently k) is small, then an attacker can guess c and extract x from $F(c, x)$.

It is helpful to describe these ideas in terms of a geometric analogy. Let C be the set of points on the lattice $\{100u, 100v\}$ for integer values u and v. Let us think of the witness x as a point on the Euclidian plane, say, $(745, 260)$. Let the decoding function f map a given point to the nearest lattice point in C. E.g., $f(120, 94) = (100, 100)$. Suppose we choose an arbitrary lattice point, say, $c = (300, 300)$. We can express x in the form $x = c + \delta$ by letting $\delta = (445, -40)$.

Suppose now that without knowing the codeword c, we are given the blob $y = (h(c), \delta)$. (This y, as we shall see, is exactly the commitment of c.) Observe that δ tells us the position of x relative to c, but gives us no information about what c is. Thus, assuming that h is a secure one-way function, the only information that y effectively reveals about the witness x is that it takes the form $(100u' + 45, 100v' + 60)$ for some integers u' and v'. Subject to this constraint, x could otherwise lie anywhere in plane.

Consider the case, now, if we are additionally given some point x' that is close to x, say $x' = (720, 240)$. By subtracting δ, we translate x' to the region near the codeword c. In particular, $x' - \delta = (275, 280)$. By applying the decoding function f to this last point, we obtain $f(x' - \delta) = c$. Thus, knowledge of x' and use of the decoding function f enable us to determine x from y and decommit c.

Say that x were the fingerprint template of a user. Then an attacker with knowledge of y alone would be unable to find a witness to decommit c. On the other hand, as demonstrated above, if the user were to present her finger to a reading device, generating read data x', then it would be possible to extract c from y. It is easy to see, in consequence, that knowledge of y makes it possible to verify that x' is close to x, and thus to authenticate the user. In loose terms, x' may be viewed as a fuzzy representation of the original witness x. Let us proceed to make this intuition more precise.

4.2 Construction of F

Our construction for F is now quite straightforward. Let $h : \{0,1\}^n \to \{0,1\}^l$ be a hash (or one-way) function such as, e.g., SHA-1. We now formally define $F : (\{0,1\}^n, \{0,1\}^n) \to (\{0,1\}^l, \{0,1\}^n)$ as follows:

$$F(c, x) = (h(c),\, x - c).$$

To decommit $F(c, x) = (\alpha, \delta)$ using witness x', the receiver computes $c' = f(x' - \delta) = f(c + (x' - x))$. If $\alpha = h(c')$, then the blob has been successfully decommitted, with c' representing the extracted commitment. Otherwise, x' is an incorrect witness. Provided f is an efficient decoding function (which is the case, of course, for codes used in practice), then decommitment is likewise an efficient process. In the remainder of the paper, we shall denote the entire commitment scheme, both the commitment and decommitment processes, informally by F.

Recall that the "fuzziness" of F consists of the notion that if x' is close to x, then x' can be used to decommit $F(c, x)$. This notion formalized in the following lemma, whose proof is given in the appendix. Note that the converse does not necessarily hold.

Lemma 1 *Suppose that $\| x - x' \| \leq t$. Then for any c, the witness x' can be used to decommit $F(c, x)$ successfully.* ▌

4.3 Applications of the fuzzy commitment function F

To provide a flavor of how a fuzzy commitment scheme might be deployed in a biometric systems, we now sketch how F can be used to achieve three different security goals, namely static (or off-line) authentication, challenge-response authentication, and encryption/decryption. We assume that a user presents a secret x in an enrollment (or encryption) phase and in any given subsequent interaction presents some x' that, if legitimate, differs from x by at most the correction threshold t. In a biometric system, once again, x might be the fingerprint template presented by the user in an enrollment phase. In this case, x' is fingerprint information presented for authentication at the initiation of a login session. We use $\in_R$ in what follows to denote uniform random selection from a set.

In all three protocols, the basic idea is the same. The witness x is used to commit to a secret codeword c. Presentation of a witness x' close to x opens this secret c, which may then be used to achieve the desired security goal, be it encryption, decryption, or authentication. Note as mentioned above, however, that in the first authentication protocol, the committed value c does not play a direct role as a cryptographic key. It must nonetheless be selected from a large space C in order to ensure that x remains well concealed, as well as to achieve a sufficiently high level of security in the authentication scheme.

Fuzzy authentication Let S denote the authentication entity, such as a server verifying biometric data to control resource access. Let U denote the user. Our protocol is as follows.

- **Enrollment** The user U presents biometric data x. The authentication system S selects a codeword $c \in_R C$. S computes the commitment $y_U = F(c, x)$, and stores it in a file for user U. Alternatively, for off-line applications, it is possible to store y_U and a digital signature of S on y_U in, say, a smart card.
- **Authentication** A user purporting to be U presents a value x' for authentication. S looks up y_U and checks whether the witness x' yields a

successful decommitment. If so, the user is authenticated as U; otherwise, the authentication fails. The authentication may, alternatively, take place off-line in some trusted module.

Note also that the length of the authentication data y_U is just $n + l$ bits, the length of the value x plus the length of the image of h. For a standard hash function like SHA-1, the fuzzy commitment of a biometric template is only 20 bytes longer than the template itself.

The following small example is intended to provide some flavor of how authentication might work under a fuzzy commitment.

Example 2 *Let us extend our simple zero-one-block code of Example 1 and consider its application to a toy fingerprint authentication system in which* $n = 10$. *Let* C *consist of the set of four codewords* $\{00000, 11111\}^2$. *Let* f *perform majority error-correction sequentially on blocks of five bits in the obvious way. Observe that* $t = 2$ *for this code.*

Suppose that a user enrolls a fingerprint template $x = 01010\ 10101$ *in an authentication system (the space in the representation of* x *is inserted here for clarity). Suppose further that the system randomly chooses the codeword* $c = 00000\ 11111$. *Thus,* $\delta = 01010\ 01010$. *The authentication system should store the commitment* $F(c, x) = (\alpha, \delta) = (h(00000\ 11111),\ 01010\ 01010)$.

Now suppose that when the user goes to authenticate herself, she presents fingerprint data $x' = 11010\ 11101$. *Observe that the value* x' *differs from* x *in two bit positions. Now* $h(f(x' - \delta)) = h(f(10000\ 10111)) = h(00000\ 11111) = \alpha$. *As the decommitment is succesful, the authentication succeeds.* ▮

Fuzzy challenge-response authentication protocol F can serve as the basis of a fuzzy challenge-response authentication using any public key cryptosystem. Let K be a deterministic algorithm that takes as input a seed and outputs a corresponding secret/public key pair (SK, PK). Let $D_{SK}(m)$ denote the decryption (signature) of a message m using secret key SK, and let $E_{PK}(\Sigma)$ denote the encryption (verification) using public key PK of a message (signature) Σ. The protocol is as follows.

- **Enrollment** The User selects a codeword $c \in_R C$. She computes $F(c, x)$ and $(SK_U, PK_U) = K(c)$. She stores $F(c, x)$, and registers the key PK_U with S.

- **Authentication** The authentication entity S sends the user a random message m. The user takes data x' and tries to decommit $F(c, x)$. If successful, she uses the secret c as a seed to K to derive (SK_U, PK_U). She then produces the digital signature $D_{SK_U}(m)$ and sends it to S, who verifies that the signature is valid under public key PK_U.

Fuzzy encryption Let $E_w(m)$ denote encryption under a symmetric encryption algorithm of message m using key w. We have the following encryption algorithm based on use of fuzzy commitment.

- **Encryption** The User selects a codeword $c \in_R C$. She encrypts message m as $(E_c(m), F(c,x))$.
- **Decryption** To decrypt using x', the user first seeks to decommit $F(x,c)$ using witness x'. If successful, she extracts the encryption/decryption key c, which she can use to recover the plaintext m.

Fuzzy encryption allows applications such as that in which a user employs a fingerprint as a secret enabling encryption and decryption of files.

5 Security and Resilience

In this section, we investigate the security of our fuzzy commitment function construction. To simplify our analysis, we assume that the witness x is drawn uniformly at random from $\{0,1\}^n$. Also in this section, we consider the resilience of F. As the the reader shall see, the resilience of F is complementary, i.e., inversely related, to its level of concealment.

5.1 Security

Recall that the security of a commitment scheme consists of two properties: it must be both concealing and binding. The following theorem characterizes the property of concealment in F. The proof and some discussion may be found in the appendix.

Theorem 1 *Suppose that for $c \in_R C$ and $x \in_R \{0,1\}^n$ an attacker is able to determine c from $F(c,x)$ in time T with probability $p(T)$. Then it is possible for the attacker to invert $h(z)$ on a random input $z \in_R C$ in time T with probability $p(T)$.*

As $|C| = 2^k$, Theorem 1 indicates that k is a security parameter governing the concealment of our construction. For most applications, a value of about $k = 80$ should provide an adequate level of security. Under common assumptions about hash functions – in, e.g., the random oracle model – this security level will require from an attacker seeking to open a commitment under F an average of 2^{79} hash function computations. This is comparable to the computational effort required for factoring RSA-1024 or finding a collision in SHA-1.

Recall that the notion of *binding* in a commitment scheme conventionally refers to the property whereby it is infeasible for any polynomially bounded player to produce valid decommitments of $F(c,x)$ for two distinct values c and c'. For our scheme, we consider a strictly stronger notion of binding. We say that F is *strongly binding* if it is infeasible for any polynomially bounded player to produce a *witness collision* on F. A witness collision is a commitment $F(x, c)$ and a pair of witnesses (x_1, x_2) both of which yield valid decommitments, but such that x_1 and x_2 are not "close". We say that x_1 and x_2 are close if $f(x_1 - \delta) = f(x_2 - \delta)$, i.e., x_1 and x_2 are close in a sense defined by the underlying error-correcting code. Observe that this definition of strong binding subsumes the conventional definition of binding.[4] In particular, it is easy to see that if F is strongly binding, then F is also binding. We now have the following claim, whose proof is straightforward. [5]

Claim 1 *F is strongly binding if h is collision resistant. In particular, suppose that an attacker is capable of finding a witness collision. Then the attacker can find a collision on h.*

The notion of strong binding is particularly useful for biometric authentication scenarios. For example, consider a situation in which an attacker is capable of finding a commitment $F(x, c)$ and two substantially different witnesses x and x', both of which yield a valid decommitment of the value c. This situation is not captured by the weaker definition of binding. In the setting of biometric authentication, however, it might correspond to a situation in which the attacker can register a pair of fingerprints from two different people that would be identified as belonging to the same person. Thus, strong binding ensures against, e.g., a repudiation attack, in which the user of security system registers two different keys and then claims his data has been compromised by a party possessing a different key. This is sometimes an important property for the applications described in Section 4.3.

Claim 1 states that the length l of images output by h dictates the security level of the strong binding property, i.e., the hardness of finding a witness collision. Under the common assumption that the most effective means of finding a collision in a hash function is a birthday attack (see [25] for definition), the induced work factor is $2^{l/2}$. Hence a security parameter of $l = 160$, which corresponds to the image length of SHA-1, yields a minimum work factor of about 2^{80}.

5.2 Resilience: What % error can F tolerate?

We now consider the tolerance of our technique to errors in the witness. Let F be a fuzzy commitment scheme and let $F(c, x) = (\alpha, \delta)$ be the commitment generated for a bitstring x with a randomly generated codeword c. We say that

[4]Strong binding may, of course, also be defined in a conventional commitment scheme by allowing a witness collision to include any x_1 and x_2 that are distinct.

[5]In contrast to Theorem 1, we do not measure the success of the attacker as a function of time here. This is due to our use of a fixed hash function, since for any given hash function h, there exists a trivial, constant-time algorithm that finds a collision. This algorithm simply outputs a known collision.

F has $q\%$ resilience for the pair (x, c) if for *error term* e such that $\parallel e \parallel \leq \frac{qn}{100}$, $x' = x + e$ is also a witness sufficient to decommit (α, δ). If F has $q\%$ resilience for all pairs of bitstrings and codewords (x, c), we say simply that F is $q\%$ *resilient.*

The resilience of a fuzzy commitment scheme is easily seen to be bounded below by the resilience of the error-correcting code used in its construction. If the code itself has a correction threshold of $\frac{qn}{100}$, then F is $q\%$ *resilient.* This follows since by definition $f(c + e) = f(c)$ for any codeword c and any error term e such that $\parallel e \parallel \leq \frac{qn}{100}$. Thus for any bitstring x and codeword c, we have $h(f(x + e - \delta)) = h(f(x + e - (x - c))) = h(f(c + e)) = h(f(c)) = \alpha$, so that $x + e$ will decommit (α, δ).

As remarked above, the correction threshold of an error-correcting code is bounded by the minimum Hamming distance between codewords in C (known as the *minimum distance* of the code). In general, the larger the coding efficiency k/n, the larger the minimum distance achievable in an error-correcting code. (This is logical, as k/n is proportional to the redundancy permitted in the code.) Often, however, we do not have much control over the values n and k. As detailed in our security analysis, k should be approximately 80 to prevent brute-force inversion attacks against the underlying hash function h. The value n is typically fixed by the particular application.

For fixed parameters k and n, there is no straightforward way to determine the most efficient error-correcting code. The design of codes to handle particular parameter sets is a broad research topic covered in some degree by classic texts such as [23] or [28]. In general, practicioners resort to tables of the best known codes, such as those given in [28].

To provide some sense of the level of resilience achievable in practical settings, however, let us consider the case where $n = 540$. This corresponds to a rough lower bound on the amount of information in a typical template extracted by the latest generation of fingerprint scanning chips manufactured by Veridicom [22]. Consulting the table in [28] on an efficiently computable class of error-correcting codes known as BCH codes, we find that a BCH code exists with $k = 76, n = 511$ and a correction threshold of 85 bits. The parameter $k = 76$ provides an acceptable security level, and we can use codewords of length 511 by truncating or compressing some data. This BCH code enables us to construct a fuzzy commitment scheme that tolerates errors in any witness of up to almost 17% of the component bits.

5.3 Modifying distribution assumptions

Non-uniform distributions on witness x We have assumed throughout our exposition above that witnesses x to the commitment scheme are selected uniformly at random from $\{0, 1\}^n$. If this is not the case, and x is drawn from some non-uniform distribution D over $\{0, 1\}^n$, then Theorem 1 no longer holds. Some distributions D will not result in a significant diminution in the security parameter k, while others will yield a lesser security level. A good security analysis will, in general, require detailed knowledge of D. On the other hand, if D is only slightly non-uniform, then it is straightforward to show that only a slight diminution in security will result. Larger diminutions in security can be compensated for by increasing k (and thereby possibly reducing the resilience of the commitment scheme).

Beating the correction threshold The error term $e = x' - x$ will, in a biometric system, typically represent the difference between a biometric template and data presented during an authentication. In many cases, the bits in e are distributed independently. In other words, the corrupted witness x' results from the addition of noise that alters every bit of x independently with some probability p. In this case, it is generally not possible to achieve resilience much better than the correction threshold t for the error-correcting code. On the other hand, if bits in e are correlated, then we can sometimes construct codes that achieve higher level of resilience that the correction threshold. This is because correlations in e restrict the number of likely error patterns. If errors tend to occur in sequence, for example, then it is advantageous to use Reed-Solomon codes, well-known for their use in the digital recording media such as compact discs, where so-called burst errors are common [36]. An additional advantage of Reed-Solomon codes is that for this class of code much progress has been made recently in achieving probable error correction beyond the correction threshold [29, 34, 35]. In certain cases, it may even be possible to use such codes to achieve good error correction under independence of bits in e.

Real-world biometric systems Regrettably, a rigorous characterization of the typical error level in the Veridicom and other fingerprint readers is not yet available. The error level and typical input distributions for some readers, such as the iris scanner of IrisScan™, are better understood (see, e.g., [15]), but not sufficiently for a good analysis of their potential for secure error correction. The distribution characteristics for biometric readers on typical human population segments represents an important research topic.

Another important research topic treats the conversion of biometric templates to bitstrings or other representations amenable to fuzzy commitment. While IrisScan™ and some other biometric templates take the form of bitstrings, many fingerprint image templates do not. Pattern matching methods that involve conversion from native to more conventional representations, however, are an active area of research [22]. In order to apply our fuzzy commitment scheme with firm security guarantees to existing biometric systems, it may be necessary to await advances in this area, as well as in characterization of template structures.

6 Conclusion

We have constructed a simple and practical fuzzy commitment scheme using well-known techniques from error-correcting codes and cryptography. Our work prompts a number of further questions. Foremost is the question of the distribution of inputs in biometric authentication and other real-world applications. Are there common biometric template types that are uniformly or near uniformly distributed? If not, can our fuzzy commitment function construction be adapted to provide strong security guarantees? Also important is the question of what types of error patterns are common in real-world applications and, consequently, what error-correcting codes are most suitable. (It is our suspicion that recent research on Reed-Solomon codes may provide useful results in this area.) A final avenue of exploration is to find new applications of fuzzy commitment schemes, perhaps to such areas as multimedia transmission over noisy channels or digital watermarking.

Acknowledgments

It has been brought to the attention of the authors that an idea similar to that presented here was developed by a research team at Sandia several years ago, but could not be published because of secrecy restrictions. We wish to acknowledge Yair Frankel, one of the co-inventors, for apprising us of this prior discovery, as well as for his comments and support. The authors wish also to thank Geoff Gordon, Markus Jakobsson, Burt Kaliski, Larry O'Gorman, Amin Shokrollahi, and Lisa Yin for their advice and comments.

References

[1] M. Alabbadi and S.B. Wicker. A digital signature scheme based on linear error-correcting block codes. In Josef Pieprzyk and Reihanah Safavi-Naini, editors, *Advances in Cryptology - ASIACRYPT '94*, pages 238–248. Springer-Verlag, 1994. LNCS No. 917.

[2] B. DePalma, Director. Mission: Impossible. Paramount Pictures, 1997. Starring Tom Cruise *et al.*

[3] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On password-based authenticated key exchange using collisionful hash functions. In *The Australian Conference on Information Security and Privacy (ACISP '96)*, pages 299–310, 1996. LNCS No. 1172.

[4] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On selectable collisionful hash functions. In *The Australian Conference on Information Security and Privacy (ACISP '96)*, pages 287–292, 1996. LNCS No. 1172.

[5] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On the weaknesses of Gong's collisionful hash function. *Journal of Universal Computer Science (J.UCS)*, 3(3):185–196, 1997.

[6] C.H. Bennett, F. Bessette, G. Brassard, G. Savail, and J. Smolin. Experimental quantum cryptography. *Journal of Cryptology*, 5(1):3–28, 1992.

[7] C.H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. Practical quantum oblivious transfer protocols. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, pages 351–366. Springer-Verlag, 1991. LNCS No. 576.

[8] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24:384–386, 1978.

[9] T.A. Berson, L. Gong, and T.M.A. Lomas. Secure, keyed, and collisionful hash functions. Technical Report SRI-CSL-94-08, Computer Science Laboratory, SRI International, December 1993.

[10] W. Branigin. INS fighting for a high-tech future. *Washington Post*, page A19, 30 September 1997.

[11] R. Chandrasekaran. Brave New Whorl: ID systems using the human body are here, but privacy issues persist. *Washington Post*, page HO 1, 30 March 1997.

[12] D. Chaum, I.B. Damgård, and J. van de Graaf. Multiparty computation ensuring privacy of each party's input and correctness of the result. In C. Pomerance, editor, *Advances in Cryptology - CRYPTO '87*, pages 87–119. Springer-Verlag, 1987. LNCS No. 293.

[13] C. Crépeau. Efficient cryptographic protocols based on noisy channels. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, pages 306–317. Springer-Verlag, 1997. LNCS No. 1233.

[14] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *Proceedings of the 29th IEEE Symposium on the Foundations of Computer Science*, pages 42–52, 1988.

[15] J. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):648–656, November 1993.

[16] G.I. Davida, Y. Frankel, and B.J. Matt. On enabling secure applications through off-line biometric identification. In *IEEE Symposium on Privacy and Security*, 1998. To appear.

[17] G.I. Davida, Y. Frankel, and B.J. Matt. On the relation of error correction and cryptography to an offline biometric based identification scheme. In *Proceedings of WCC99, Workshop on Coding and Cryptography*, 1999. To appear.

[18] D.C. Feldmeier and P.R. Karn. UNIX password security – ten years later. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89*, pages 44–63. Springer-Verlag, 1989. LNCS No. 435.

[19] R. Fixmer. Tiny new chip could pit protection of property against right of privacy. *New York Times*, 23 September 1998.

[20] L. Gong. Collisionful keyed hash functions with selectable collisions. *Information Processing Letters*, 55(3):167–170, August 1995.

[21] T. Jakobsen. Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, pages 212–222. Springer-Verlag, 1998. LNCS No. 1462.

[22] L. O'Gorman, Chief Scientist, Veridicom Corp., 23 September 1998. Personal communication.

[23] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier, 1977.

[24] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical Report DSN progress report 42-44, Jet Propulsion Laboratory, Pasadena, 1978.

[25] A.J. Menezes, S.A. Vanstone, and P.C. van Oorschot. *Handbook of Applied Cryptography*. CRC Press, 1996.

[26] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 22:594–597, 1979.

[27] N. Meyer, Director. Star Trek II: The Wrath of Khan. Paramount Pictures, 1982. Starring William Shatner *et al.*

[28] W.W. Peterson and E.J. Weldon, Jr. *Error-Correcting Codes, Second Edition*. MIT Press, 1972.

[29] M.A. Shokrollahi and H. Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. In *The Thirtieth Annual ACM Symposium on Theory of Computing (STOC '98)*, 1998. To appear.

[30] C. Soutar. Biometric encryption for secure key generation, January 1998. Presentation at the 1998 RSA Data Security Conference.

[31] C. Soutar and G.J. Tomko. Secure private key generation using a fingerprint. In *CardTech/SecurTech Conference Proceedings, Vol. 1*, pages 245–252, May 1996.

[32] J. Stern. A new identification scheme based on syndrome decoding. In D.R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, pages 13–21. Springer-Verlag, 1993. LNCS No. 773.

[33] D. Stinson. *Cryptography: Theory and Practice.* CRC Press, 1995.

[34] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. Also published in FOCS '96 under the title "Maximum likelihood decoding of Reed Solomon Codes".

[35] M. Sudan and V. Guruswami. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS '98)*, 1998. To appear.

[36] S.A. Vanstone and P.C. van Oorschot. *An Introduction to Error Correcting Codes with Applications.* Kluwer Academic Publishers, 1989.

[37] L.A. Zadeh, R.R. Yage (Editor), R.R. Yager, R.M. Tong (Editor), and H.T. Nguyen (Editor). *Fuzzy Sets and Applications : Selected Papers by L.A. Zadeh.* John Wiley & Sons, 1987.

A Proofs

Lemma 1 *Let F be a fuzzy commitment scheme based on an error-correcting code with error-correcting threshold t. Suppose that $\| x - x' \| \leq t$. Then for any c, the witness x' can be used to decommit $F(c, x) = (\alpha, \beta)$.*

Proof: Since t is the correction threshold of the code C, for any $e \in \{0,1\}^n$ with $\| e \| \leq t$ we have $f(c + e) = c = f(c)$. Since $\| x - x' \| \leq t$, it follow that for any bitstring x and codeword c, we have $h(f(x' - \beta)) = h(f(x' - (x - c))) = h(f(c + x' - x)) = h(f(c)) = \alpha$, so that x' will decommit (α, β). ▮

Theorem 1 *Suppose that for $c \in_R C$ and $x \in_R \{0,1\}^n$ an attacker is able to determine c from $F(c, x)$ in time T with probability $p(T)$. Then it is possible for the attacker to invert $h(z)$ on a random input $z \in_R C$ in time T with probability $p(T)$.*

Proof: Since x and c are selected independently and uniformly at random, it is clear that $\delta = x - c$ reveals no information about the codeword c. It follows that the task of an attacker in determining c is equivalent to the task, given knowledge only of $h(c)$, of finding a string $z \in C$ such that $h(z) = h(c)$. The theorem follows. ▮

Remark The underlying assumption in Theorem 1, that it is hard to invert h on images drawn from C, is somewhat non-standard. It is in accordance, though, with common security assumptions on hash functions, such as those provided by the random oracle model. Nonetheless, we can easily recast Theorem 1 to use more canonical security assumptions. For any $c \in C$, let $\tilde{h}(c) = h'(g^{-1}(c))$, where $h' : \{0,1\}^k \to \{0,1\}^k$ is a one-way permutation. (Recall here that g^{-1} is a one-to-one function that maps a codeword to its corresponding message in M.) If we substitute $\tilde{h}$ for h in our construction of F, then the security of F relies on the hardness of inverting the one-way permutation h' on a random image. Theorem 1 can be modified accordingly to rely on this more standard security assumption.