A Fuzzy Vault Scheme

Ari Juels[1] and Madhu Sudan[2]

[1] RSA Laboratories
Bedford, MA 01730, USA
E-mail: ajuels@rsasecurity.com
[2] MIT Laboratory for Computer Science
200 Technology Square, Cambridge, MA 02139
E-mail: madhu@mit.edu

Abstract. We describe a simple and novel cryptographic construction that we refer to as a *fuzzy vault*. A player Alice may place a secret value κ in a fuzzy vault and "lock" it using a set A of elements from some public universe U. If Bob tries to "unlock" the vault using a set B of similar length, he obtains κ only if B is close to A, i.e., only if A and B overlap substantially. In constrast to previous constructions of this flavor, ours possesses the useful feature of *order invariance*, meaning that the ordering of A and B is immaterial to the functioning of the vault. As we show, our scheme enjoys provable security against a computationally unbounded attacker.

Keywords: biometrics, error-correcting codes, fuzzy commitment, fuzzy vault

1 Introduction

Alice is a movie lover. She is looking to find someone who shares her taste in movies, but does not want to reveal information about her preferences indiscriminately to other people. One approach she might take is to compile a set A of her favorite movies and publish it in a concealed form. For instance, Alice might post to a Web newsgroup a ciphertext C_A representing an encryption of her telephone number tel_A under the set (here, key) A. In this case, if another person, say Bob, comes along with a set B of his own favorites that is identical to A, then he can decrypt C_A and obtain Alice's phone number. If Bob tries to decrypt C_A with a set different than Alice's, he will fail to obtain her telephone number. A drawback to this approach is its exactitude, or lack of error-tolerance. If Bob's interests are very similar to Alice's, e.g., if he likes two or three films that Alice doesn't, then he will not learn tel_A. It seems very likely in this case, though, that Alice would still like Bob to obtain her telephone number, as their tastes are quite similar.

In this paper, we introduce the notion of a *fuzzy vault*. This is a cryptographic construction whereby Alice can *lock* her telephone number tel_A using the set A, yielding a *vault* denoted by V_A. If Bob tries to *unlock* the vault V_A using his own set B, he will succeed provided that B overlaps largely with A. On the other hand, anyone who tries to unlock V_A with a set of favorite movies differing substantially from Alice's will fail, helping to ensure that Alice's set of favorites remains private. Thus, a fuzzy vault may be thought of as a form of error-tolerant encryption operation where keys consist of sets. Our fuzzy vault proposal has two important features that distinguish it over similar, prior work. First, the sets A and B may be arbitrarily ordered, i.e., true sets rather than sequences. Second, in contrast to previous work, we are able to prove information-theoretic security bounds over some natural non-uniform distributions on the set A.

Error-tolerant cryptographic algorithms are useful in many circumstances in which security depends on human factors, and thus exactitude represents a drawback. We offer just a few examples here, all of which might benefit from use of our fuzzy vault scheme:

1. **Privacy-protected matching:** As an extension of our movie lover's example above, we might consider a business scenario. Bisco Corp. is looking to sell routers possessing a set $A = \{a_1, a_2, \ldots, a_k\}$

of specifications. It might publish a fuzzy vault V_A with its identity κ, locked under A. If Disco Corp. is looking for routers with a set B of similar specifications, then it will be able to open the vault. Anyone who tries to unlock the vault with a dissimilar set will not learn κ. (We address this idea in detail later in the paper, and decribe an important security enhancement using on-line throttling mechanisms.)

2. **Personal entropy systems:** Proposed by Ellison *et al.* [10], this is a system that enables users to recover passwords by answering a series of questions. In recognition of the unreliability of human memory, the system permits users to answer some of these questions incorrectly. A serious vulnerability in this system is exposed in [4], who show more broadly that the underlying hardness assumption is weak. Our fuzzy vault scheme offers an alternative implementation that is provably secure in an information-theoretic sense and that may involve use of sets, and not just fixed-order answers.
3. **Biometrics:** Alice authenticates to her server using fingerprint information. Her system administrator wishes to store her fingerprint on the server or, more precisely, a set A of features characterizing her fingerprint. (Such sets are known as biometric *templates*.) If an attacker breaks into the server, however, Alice does not want her template A compromised. An additional complication is that biometric systems are error-prone: When Alice tries to authenticate herself, her fingerprint reader is likely to produce a template A' that is similar to, but not identical to A (with bit errors and random permutation and erasure of elements). Alice might store a PIN locked in a fuzzy vault on a set A of features describing her fingerprint, thereby achieving both error-tolerance and privacy. Note that order-invariance is critical here. It is usually not possible to impose an order effectively on biometric features because of the problem of erasures. For this reason, previous schemes like that of Juels and Wattenberg [15] described below are unlikely to work well for this problem.

1.1 Previous work

A somewhat less naïve approach to a fuzzy vault construction than straightforward encryption might be achieved through use of Shamir secret sharing techniques [22]. Alice partitions her secret value κ into shares $s_1, s_2, \ldots, s_n$, and encrypt these shares respectively under each of the elements $a_1, a_2, \ldots, a_n$ in her set A. This would yield a set of ciphertexts $e_1, e_2, \ldots, e_n$. Given use of a (t, n)-secret sharing scheme, Bob would only need to decrypt t shares successfully in order to unlock Alice's secret κ. The problem with this approach is twofold. First, suppose that Bob's set B consists of elements $b_1, b_2, \ldots, b_n$. Because A and B are unordered sets, Bob has no way of knowing which of the ciphertexts e_i to try to decrypt with a given set element b_j. Even if Bob tries all n^2 possible decryption operations, there is a second problem: He still does not know which decryptions were successful. Straightforward mechanisms to reveal this information to Bob leak substantial information about A. Indeed, this may be regarded as the source of the weakness in, e.g., the Ellison *et al.* construction. It is also possible for Bob to try to deduce by means of a brute-force search which elements of B do not overlap with those of A. This strategy is inflexible and likely to be prohibitively slow in many practical scenarios, as the computational requirements grow exponentially in the size of $|A \bigcap B|$.

Another idea that does not work well is that of imposing a common ordering on the sets A and B and then using a fuzzy vault construction or similar technique that does not have the property of order invariance, e.g., [15]. This appeoach fails because a small difference between sets can produce large differences in an ordered element-by-element comparison. Suppose, for example, that A and B again represent respective lists of Alice and Bob's favorite movies. If Alice and Bob's favorites include all Oscar winners, except that Alice does not like *Antonia's Line*, then a movie-by-movie comparison of these lists in alphabetical order will yield almost no matches, while in fact A and B overlap considerably. This problem also applies to attempts to impose ordering on features in biometric systems.

To overcome these difficulties, we invoke *error-correcting codes* as the basis for our construction. Given the strong technical and historical affinities between error-correcting codes and cryptographic codes, it is not surprising that error-correcting codes appear in many areas of cryptography, such as quantum cryptography [2, 6], public-key cryptography (via the well known McEliece cryptosystem) [17], identification schemes [25], digital signature schemes [1], and cryptanalytic techniques [13], just to name a few examples. We do not explore this extensive branch of the literature here. We note, however, that Reed-Solomon codes, the most popular form of error-correcting code and the one we focus on here, may be viewed as a general, error-tolerant form of Shamir secret sharing.

The starting point for our fuzzy vault construction is the *fuzzy commitment* scheme of Juels and Wattenberg [15], which is also based on the use of error-correcting codes. This is a cryptographic primitive whereby a user commits to a secret value κ under a key x. The user may decommit using any key x' that is "close" to x under some suitable metric, such as Hamming distance. An attacker without any knowledge of x, however, cannot feasibly decommit κ. One application of fuzzy commitment, as suggested by the authors, is to securing biometric systems, as described above. An enrolled fingerprint image (known as a *template*), for example, might be viewed as a key x. The user tries to authenticate using another, slightly different image of the same finger, which we may denote by x'. Authentication is successful if and only if x' is "close" to x.

As the fuzzy commitment scheme in [15] is antecedent to our own, it is worth briefly sketching the details. Let $\mathcal{F}$ be a field, and C be the set of codewords for some error-correcting code; assume that codewords lie in $\mathcal{F}^n$. To commit to a value $x \in \mathcal{F}^n$, the user selects a codeword c uniformly at random from C and computes an offset of the form $\delta = c - x \in \mathcal{F}^n$, i.e., the difference over individual field elements. The commitment then consists of the pair (δ, y), where $y = h(c)$ for some suitable one-way function h. To decommit using key x', the user computes $\delta + x'$ and, if possible, decodes to the nearest codeword c'. The decommitment is successful iff $h(c') = y$.

The construction in [15] has the advantageous features of conceptual simplicity and the ability to make use of any underlying error-correcting code. Moreover, provided that x is drawn uniformly at random from $\mathcal{F}^n$, the scheme enjoys rigorously proveable security linear in the cardinality of C. Suppose that the attacker gains no information about c or x from y, as would be the case under a random oracle assumption on h given sufficiently large security parameters. It is easy to see then that the task of the attacker is to guess c uniformly over C. A similar, less resilient antecedent scheme is proposed in [7, 8], while another system with similar goals but no rigorously provable security characteristics is proposed in [23, 24].

Note that if the hashed value $h(c)$ is removed from the Juels and Wattenberg scheme, i.e., if we no longer think of it as a commitment scheme, then we obtain a kind of fuzzy vault in which the vault itself is equal to δ. If x is uniformly distributed, then the scheme enjoys easily provable information-theoretic security, i.e., security against a computationally unbounded attacker (also proportional to the cardinality of C). Like our own fuzzy vault construction, this one can also be applied to any of the three practical scenarios described above, i.e., privacy-protected matching, personal entropy systems, or biometrics.

As a fuzzy vault variant, though, the scheme of Juels and Wattenberg has two shortcomings. First, while it tolerates some number of errors in the information symbols in x, it does not tolerate substantial re-ordering of these symbols. Given that translation and rotation errors are common in, e.g., biometric systems, it is reasonable to expect that the image x' may consist of a permutation of symbols in x. The property of *order-invariance* is thus likely to be desirable in a fuzzy commitment scheme. A second shortcoming of [15] is the difficulty of proving rigorous results about security over non-uniform distributions. Our proposed scheme addresses these two shortcomings, and may be thought of as an order-invariant version of the Juels-Wattenberg scheme.

1.2 Our scheme

Like the scheme of Juels and Wattenberg, ours is conceptually simple, and can be implemented using any underlying error-correcting code (although we focus on Reed-Solomon codes in our exposition here). While possessing the advantages of order-invariance and easier analysis on non-uniform distributions, our scheme does have a couple of drawbacks that are important to note from the outset. First, it typically has substantially greater – though still quite practical – memory requirements than the Juels-Wattenberg scheme. Second, it is somewhat less flexible in terms of available parameter choices at a given security level, as we shall see.

Let us briefly sketch the intuition behind our scheme. Suppose that Alice aims to lock a secret κ under set A. She selects a polynomial p in a single variable x such that p encodes κ in some way (e.g., has an embedding of κ in its coefficients). Treating the elements of A as distinct x-coordinate values, she computes evaluations of p on the elements of A. We may think of Alice as projecting the elements of A onto points lying on the polynomial p. Alice then creates a number of random *chaff* points that do not lie on p, i.e., points that constitute random noise. The entire collection of points, both those that lie on p and the chaff points, together constitute a commitment of p (that is, κ). Call this collection of points R. The set A may be viewed as identifying those points in R that lie on p, and thus specifying p (and κ). As random noise, the chaff points have the effect of concealing p from an attacker. They provide the security of the scheme.

Suppose now that Bob wishes to unlock κ by means of a set B. If B overlaps substantially with A, then B identifies many points in R that lie on p, so that Bob is able to recover a set of points that is largely correct, but perhaps contains a small amount of noise. Using error correction, he is able to reconstruct p exactly, and thereby κ. If B does not overlap substantially with A, then it is infeasible for Bob to learn κ, because of the presence of many chaff points. (If B overlaps "somewhat", then he may still be able to recover κ. The gap between feasible recovery and infeasible is fairly small, however, as we discuss below.) We present details and analysis in the body of the paper.

The hardness of our scheme is based on the *polynomial reconstruction* problem, a special case of the Reed-Solomon list decoding problem [4]. Other schemes making use of this problem include, for example, the scheme proposed by Monrose, Reiter, and Wetzel for hardening passwords using keystroke data [18]. An important difference between our scheme and previous ones of this flavor is our range of parameter choices. The [18] scheme bases its security on the computational hardness of small polynomial reconstruction instances, while we select parameters enabling us to achieve information theoretic security guarantees for the same problem.

Organization

We sketch protocol and security definitions for our scheme in section 2. In section 3, we present protocol details for our fuzzy vault scheme. We offer security analysis in section 4, and conclude briefly with some implementation ideas in section 5. Security proofs are included in the paper appendix.

2 Definitions and Background

We work over a field $\mathcal{F}$ of cardinality q and a universe $\mathcal{U}$; for convenience, we assume in our exposition that $\mathcal{U} = \mathcal{F}$, although this need not be the case in generally. Our aim is to lock a secret value $\kappa \in \mathcal{F}^k$ under a secret set $A \in \mathcal{U}^t = \mathcal{F}^t$, for protocol parameters k and t. We consider a fuzzy vault algorithm Lock that takes as input a secret κ and set A and outputs a vault $V_A \in \mathcal{F}^r$ for some security parameter r. The algorithm Lock may be (and for our purposes will be) probabisetic.

A corresponding decryption algorithm Unlock takes as input a vault $V_A \in \mathcal{F}^r$ and a decryption set $B \in \mathcal{U}^t$. The output of this algorithm is a plaintext value $\kappa' \in \mathcal{F}^k$, or else $'null'$ if the algorithm is unable to extract a plaintext.

Our goal is to come up with a pair of vault locking/unlocking algorithms Lock/Unlock that allows reconstruction of the plaintext κ when the decryption set B is close to the encryption set A. At the same time, we want the vault V_A not to reveal κ. Recall from above that we are interested in algorithms that are order invariant. In other words, the ordering on the sets A and B should have no real impact on the locking and unlocking procedures.

2.1 Requirements

The next three definitions formalize the requirements of a good pair (Lock, Unlock) of algorithms for our fuzzy vault scheme. We say that a probability is *negligible* if it is asymptotically smaller than any positive polynomial in t and r. We say that a probability is *overwhelming* if it is larger than $1-\zeta$ for some negligible quantity ζ. Our first definition outlines the completeness condition for our algorithms, i.e., what should happen when the players are honest.

Definition 1. An locking/unlocking pair (Lock, Unlock) with parameter set (k, t, r) is *complete* with ϵ-fuzziness if the following holds. For every $\kappa \in \mathcal{F}^k$ and every pair of sets $A, B \in \mathcal{U}^t$ such that $|A - B| \leq \epsilon$, it is the case that $\text{Unlock}(B, \text{Lock}(A, \kappa)) = \kappa$ with overwhelming probability.

We now formalize the security, and in particular the soundness of the algorithmic pair (Lock, Unlock) in an information-theoretic sense. Assume that A is selected according to some potentially non-uniform distribution d. We seek to characterize the ability of an attacker with unlimited computational power to determine κ from $\text{Lock}(A, \kappa)$. We assume that this attacker is given knowledge of a uniformly random δ-fraction of A, i.e., a random subset A' of at most δt elements in A (where we assume δt to be an integer). This assumption that the adversary has knowledge of part of the secret key A is slightly unorthodox. In a "fuzzy" system, however, it is natural to consider such notions of partial adversarial knowledge, as highlighted in our examples below. Of course, other security assumptions are possible.

We characterize security in terms of the following experiment with a computationally unbounded adversary Adv for a given parameter set. This adversary Adv takes as input a set of δt elements of A, the parameters t and k, and a vault V_A on A, and outputs a guess at κ. Formally, Adv is an algorithm $Adv : \mathcal{U}^{\delta t} \times Z^2 \times \mathcal{F}^r \rightarrow \mathcal{F}^k$ with no bound on computational complexity. Let $\in_d$ denote selection from probability distribution d, and $\in_U$ denote uniform random selection. Here, and in all analysis that follows, we assume that κ is generated uniformly at random, as κ is typically used as a key for some independent ciphertext or cryptographic protocol. Let $\{A\}_i$ denote the set of subsets of A of cardinality i. The experiment is as follows.

Experiment $\mathsf{Attack}(\text{Lock},\ Adv)$
  $\kappa \in_U \mathcal{F}^k$; $A \in_d \mathcal{U}^t$; $A' \in_U \{A\}_{\delta t}$;
  if $Adv(A', t, k, \text{Lock}(A, \kappa)) = \kappa$
    $\mathsf{Output}'1'$;
  else
    $\mathsf{Output}'0'$;

This leads to the following definition.

Definition 2. An encryption/decryption pair (Lock, Unlock) is information theoretically secure with parameter pair (δ, μ) if $\mathsf{pr}[\mathsf{Attack}(\text{Lock},\ Adv) = 1] \leq \mu$ for any computationally unbounded adversary Adv.

Let d' be the probability distribution d restricted to sets A such that $A' \subset A$. Observe that given vault V_A, the best strategy a (computationally unbounded) adversary can adopt is to output a plaintext κ' such that the expression

$$w(\kappa', V_A) = pr_{A \in_{d'} \mathcal{U}^t}[\textsc{Lock}(A, \kappa') = V_A]$$

is maximized. For a given vault $V_A = \textsc{Lock}(A, \kappa)$, the probability of success of this strategy is easily seen to be $w(\kappa, V_A)/\sum_{\kappa' \in \mathcal{F}^k} w(\kappa', V_A)$.

Remark: Note that our definition of information theoretic security does not necessarily imply that the secret κ is *fully* protected in an information theoretically secure manner. In particular, we may have mutual information $I(\textsc{Lock}, \kappa) > 0$. This is to say that our scheme may offer information theoretic hiding of κ over a set of possible values smaller than $\mathcal{F}^k$.

2.2 Reed-Solomon codes

It is possible to construct a fuzzy vault scheme based on essentially any type of linear error-correcting code. To sharpen our conceptual focus and analysis, however, we restrict our attention to Reed-Solomon (R-S) codes. We are interested primarily in (k, t)-codes, i.e., those in which codewords consist of t information symbols, i.e., field elements. Each codeword corresponds to a unique polynomial p of degree less than k over $\mathcal{F}$; thus there are q^k codewords in total. In the simplest embodiment of such an R-S code, we may express a codeword as the sequence $\{y_1 = p(1), y_2 = p(2), \ldots, y_t = p(t)\}$, where $1, 2, \ldots, t$ represent the first t elements of the field $\mathcal{F}$.

If $t > k$, then a codeword may be seen to contain some redundancy. The presence of such redundancy is what permits the code to be used for error correction. Suppose that $c' = \{y'_1, y'_2, \ldots, y'_t\}$ is a *corruption* of the codeword c. In other words, we have $y'_i \neq y_i$ for some ϵ-fraction of the information symbols in c'. Provided that ϵ is small enough, the redundancy of the code is such that given just the corrupted codeword c', we can recover the original codeword c. For this, we use a *decoding algorithm* that we denote by RSDECODE. The algorithm RSDECODE takes c' as input, and provided that too much corruption has not occurred, outputs c.

The most common application of a Reed-Solomon or other error-correcting code is to message transmission over a noisy channel. For this, the procedure is as follows. The sender takes a message $\kappa \in \mathcal{F}^k$ and encodes it as a polynomial of degree at most k. The sender computes the corresponding codeword c and transmits it over a noisy channel. The noise on the channel causes a corrupted codeword c' to be obtained by the receiver. The receiver applies RSDECODE to c', obtains c, and recovers the original polynomial p and thus the message κ. As we shall see, in our scheme we may think of the noise on the channel as arising from differences between the sets A and B. By guessing A inaccurately, Bob introduces noise into the channel transmitting κ. (In contrast, the fuzzy commitment scheme in [15] never actually makes explicit use of the message space.)

2.3 Our special use of Reed-Solomon codes

For our constructions, it is convenient to consider a generalization of Reed-Solomon codes. We think of a codeword as consisting of an evaluation of a polynomial p over *any* set of t distinct points in $\mathcal{F}$. In other words, we think of a codeword as consisting of a set of pairs $\{(x_i, y_i)\}_{i=1}^t$, where $x_i \in \mathcal{F}$, all of the x_i are distinct, and $y_i = p(x_i)$.

In this generalized view, the decoding algorithm RSdecode takes as input a collection of points which are presumed to lie preponderantly on a single polynomial of pre-specified degree at most $k-1$. The RSdecode algorithm, if successful, outputs a polynomial p intersecting the large majority of input points.[3] Otherwise, the algorithm outputs $'null'$. This will happen, for instance, if no polynomial of the right degree matches the inputs adequately, or if computation of such a polynomial is too hard

[3] So-called *set decoding* algorithms may in fact produce a set of candidate polynomials. We assume that a successful algorithm outputs one of these selected uniformly at random from the entire set.

because of too much corruption of the associated codeword. The following are parameter specifics for the algorithm RSdecode.

Public parameters: A field $\mathcal{F}$.

Input: A degree parameter $k \leq q$ and a set of points $Q = \{(x_i, y_i)\}_{i=1}^{t}$ such that $x_i, y_i \in \mathcal{F}$ for $1 \leq i \leq t$.

Output: A polynomial p of degree less than k over $\mathcal{F}$, or else $'null'$. We write $\mathsf{RSdecode}(k, Q)$ to denote the output on inputs k and Q.

For our (practical) purposes, the best choice for RSdecode is generally the classical algorithm of Peterson-Berlekamp-Massey [3, 16, 20]. This algorithm decodes successfully if at least $\frac{k+t}{2}$ points in Q share a common polynomial. The best version of RSdecode to date, i.e., the one most likely to recover p successfully, is that of Guruswami and Sudan [12]. This algorithm successfully determines p provided that the number of points in Q that lie on p is at least $\sqrt{kt}$. Our preference for the classical algorithm is based on the fact that this algorithm is in general much more efficient than the Guruswami-Sudan, and has the advantage of being well studied and widely implemented. Moreover, for many of the parameter choices we are likely to encounter in practice, $\frac{k+t}{2}$ is fairly close to $\sqrt{kt}$.

3 The Fuzzy Vault Algorithms

We are now ready to detail our locking and unlocking algorithms for our fuzzy vault scheme. We first present the algorithm Lock. The basic idea here is to create a generalized Reed-Solomon codeword representing the secret κ (as a corresponding polynomial p). This codeword is computed over x-coordinates corresponding to elements in the set A. To conceal the codeword, we add chaff points, i.e., random noise in the form of random (x_i, y_i) pairs.

In our exposition here, we assume some straightforward, publicly agreed-upon method for representing the secret κ as a polynomial (e.g., taking the information symbols in κ to be the coefficients of the polynomial). We simply write $p \leftarrow \kappa$ to represent this conversion. We let $\in_U$ denote uniformly random selection from a set.

Public parameters: A field $\mathcal{F}$, a Reed-Solomon decoding algorithm RSDECODE.

Input: Parameters k, t, and r such that $k \leq t \leq r \leq q$. A secret $\kappa \in \mathcal{F}^k$. A set $A = \{a_i\}_{i=1}^{t}$, where $a_i \in \mathcal{F}$.

Output: A set R of points $\{(x_i, y_i)\}_{i=1}^{r}$ such that $x_i, y_i \in \mathcal{F}$.

Algorithm Lock

$X, R \leftarrow \phi$;
$p \leftarrow \kappa$;
for $i = 1$ to t do
  $(x_i, y_i) \leftarrow (a_i, p(a_i))$;
  $X \leftarrow X \, bigcup \, x_i$;
  $R \leftarrow R \bigcup (x_i, y_i)$;
for $i = t + 1$ to r do
  $x_i \in_U \mathcal{F} - X$;
  $y_i \in_U \mathcal{F} - \{p(x_i)\}$;
  $R \leftarrow R \bigcup (x_i, y_i)$;
output R;

So as not to leak information about the order in which the x_i are chosen, the set R may be output in a pre-determined order, e.g., points in order of ascending x-coordinates, or else in a random order. Note that chaff points in Lock are selected so as to intersect neither the set A nor the polynomial

p. This is for technical reasons, namely to simplify our security proofs. We refer to the set R and the parameter triple (k,t,r) together as a fuzzy vault, denoted by V_A.

As explained above, to unlock a vault V_A created by Alice as above, Bob tries to determine the codeword that encodes the secret κ. Recall that the set A specifies the x-coordinates of "correct" points in R, i.e., those that lie on the polynomial p. Thus, if B is close to A, then B will identify a large majority of these "correct' points. Any divergence between B and A will introduce a certain amount of error. Provided that there is sufficient overlap, however, this noise may be removed by means of a Reed-Solomon decoding algorithm. We write $\kappa' \leftarrow p$ to denote conversion of a polynomial of degree at most k to a secret in $\mathcal{F}^k$, i.e., the reverse of the procedure employed in LOCK. We let $(x_i,y_i) \overset{(b_i,\circ)}{\longleftarrow} R$ denote projection of R onto the x-coordinate b_i. In particular, if there is a pair $(b_i,y) \in R$ for any y, then $(x_i,y_i) = (b_i,y)$; otherwise a null element is assigned to the pair (x_i,y_i). Our unlocking algorithm is now as follows.

Public parameters: A field $\mathcal{F}$, a Reed-Solomon decoding algorithm RSDECODE.

Input: A fuzzy vault V_A comprising a parameter triple (k,t,r) such that $k \le t \le r \le q$ and a set R of points $\{(x_i,y_i)\}_{i=1}^r$ such that $x_i, y_i \in \mathcal{F}$. A set $B = \{b_i\}_{i=1}^t$, where $b_i \in \mathcal{F}$.

Output: A value $\kappa' \in \mathcal{F}^k \bigcup \{'null'\}$.

Algorithm UNLOCK

$Q \leftarrow \phi$;
for $i = 1$ to t do
    $(x_i,y_i) \overset{(b_i,\circ)}{\longleftarrow} R$;
    $Q \leftarrow Q \bigcup (x_i,y_i)$;
$\kappa' \leftarrow \text{RSDECODE}(k,Q)$;
output κ';

If the final decoding operation is successful, then the algorithm outputs a secret κ' which should be equal to κ if the set B is close to the original set A. If the decoding operation fails, then the algorithm outputs $'null'$.

The following proposition characterizes the completeness of our fuzzy vault scheme.

Proposition 3. *Given use of the Peterson-Berlekamp-Massey algorithm for* RSdecode*, the algorithm pair* (LOCK, UNLOCK) *above with parameter triple* (k,t,r) *is complete with* $(\frac{t-k}{2})$*-fuzziness.*

As an example of how the above algorithms might be applied, we briefly consider a parameterization of k and t in what we call the movie lover's problem, i.e., the problem described above in which Alice is seeking someone with similar taste in movies. We defer discussion of security parameters for the next section.

Example 1 (The movie lover's problem). Let us consider the movie lover's problem with a total set of 10^4 titles in which Alice selects a set A of $t = 22$ different favorites.[4] We might choose $k = 14$. Since $\frac{k+t}{2} = 18$, another movie lover with a set B of 22 favorite titles will be able to decrypt the digital box via the Peterson-Berlekamp-Massey algorithm provided that the original set A and the new set B intersect on at least 18 titles. Notice that for this choice of parameters, it is feasible to compute all possible subsets of size 18 from the set of size 22, and try interpolating from each subset. This would result, however, in an average of 3657.5 trials, while the cost of one decoding step is

[4] We consider 22 titles, as this is the number of password questions used in [10], which seems a good example application for our ideas.

easily within an order of magnitude of one interpolation step. Thus the use of RSdecode speeds up[5] the decommitment step by at least a factor of 300.

4 Security

The security of our fuzzy vault construction depends on the number of chaff points $r-t$ in the target set R. The greater the number of such points, the more "noise" there is to conceal p from an attacker. As many chaff points are added to R, there begins to emerge a set of spurious polynomials that look like p, i.e., polynomials that have degree less than k and agree with exactly t points in R. Briefly stated, the more chaff points there are, the greater the probability that some set of t of these chaff points (and/or real points) align themselves by chance on some polynomial of the desired degree. In the absence of additional information, an attacker cannot distinguish between the correct polynomial p and all of the spurious ones. Thus, p is hidden in an information-theoretically secure fashion in R, with security proportional to the number of spurious polynomials. Note that the security of the vault V_A depends exclusively on the number of such polynomials, and not on the length of the secret key κ; the vault is often weaker than the secret κ it protects (which is acceptable for the applications we describe). The following lemma proves that with high probability many polynomials degree less than k agree with the target set R in t places, i.e., that there are many spurious polynomials. This lemma and its proof are based on similar results of Dumer *et al.* [9].

Recall that the locking algorithm Lock picks t points according to a given p of degree less than k and $r-t$ random points (x_i, y_i) in $\mathcal{F} \times \mathcal{F}$ and outputs this set in random order as a vault hiding p (i.e., κ). Recall that q denotes the cardinality of $\mathcal{F}$. The following lemma is parameterized by r, k, and t and a small real number μ. Proof of the lemma may be found in the appendix.

Lemma 4. *For every $\mu > 0$, with probability at least $1-\mu$, the target set R generated by the algorithm* Lock *on polynomial p and locking set A satisfies the following condition: There exist at least $\frac{\mu}{3} q^{k-t}(r/t)^t$ polynomials p' of degree less than k such that R includes exactly t points of the form $(x, p'(x)) \in \mathcal{F} \times \mathcal{F}$.*

Example 2. As an example, consider the following choice of parameters. Suppose we pick a field of size approximately $q = 10^4$, and set $r = q$. Now let $t = 22$, i.e., the movie lovers pick twenty-two of their favorite movies out of a choice of q, and we chaff the data with $q - 22$ random points. Suppose we use this information to encrypt a polynomial of degree less than 14 (as in our earlier example). Then we expect to see about 2^{86} polynomials of degree less than 14 agreeing with 22 out of the roughly 10^4 points in R. In particular, with probability at least $1 - 2^{-43}$, there will be 2^{43} polynomials exhibiting this behavior. (Thus, we achieve what may be roughly characterized as a 43-bit security level.)

The example above suffers from a significant loss in security due to a naïve transformation of expected values to high probability results in the proof of Lemma 4. We believe that this loss in security is just an artifact of the proof, and that the true answer is perhaps more along the lines "With probability at least $1 - 2^{-83}$, there are 2^{83} polynomials agreeing with the given data on 22 points." (Thus, we get roughly 83-bit security.) Again, this conjecture remains open at this stage. For the moment, however, we try a different choice of parameters to strengthen our security analysis.

[5] Another way of viewing this is that the fuzzy vault algorithm can be enhanced by additional use of brute-force search, thereby improving the security threshold. This improvement can be made substantial without a loss of speed relative to the pure brute-force algorithm.

Example 3. Again, we pick $r = q \approx 10^4$ and $t = 22$. This time we use this information to encrypt a polynomial of degree less than 18. The decommitment works correctly with 20 agreements, and the running time is faster than a brute-force search by a factor of at least 10. Then we expect to see about 2^{139} polynomials of degree less than 18 agreeing with 22 out of the approximately 10^4 points in Q. In particular, with probability at least $1 - 2^{-70}$, there will be 2^{70} polynomials exhibiting this behavior. (Thus, we achieve what may be roughly characterized as a 70-bit security level.)

As stated above, we believe our scheme more amenable to analysis over non-uniform distributions that that in [15]. As an example, we note that the above lemma naturally adapts itself to the case where the set of locking sets A are not all considered equally likely. For simplicity we consider the case where A is equally likely to come from some family of sets $\mathcal{E} \subset 2^{\mathcal{U}} = 2^{\mathcal{F}}$, i.e., a family of sets over $\mathcal{U} = \mathcal{F}$. We have the following lemma, whose proof is provided in the appendix.

Lemma 5. *For every $\mu > 0$, with probability at least $1-\mu$, the target set R generated by the algorithm* Lock *to commit to a polynomial p with locking set A satisfies the following condition: There exist at least $\frac{\mu}{3} q^{k-t} |\mathcal{E}|$ polynomials $p' \in \mathcal{P}$ such that R agrees with p' on some subset of t points in the family $\mathcal{E}$.*

Example 4. Consider a variant of the movie lover's problem where the movie lover is expected to choose 2 movies each from 10 categories, and each category consists of 1000 movies. In this case, the distribution on movies has support on only $(\binom{10^3}{2})^{10}$ sets. The above lemma shows that with $r = 10^4$, $t = 20$ and $k = 16$, one expects to find 2^{106} polynomials of degree at most 15 agreeing with the data on 20 points, with 2 agreements each from each of 10 categories. As usual, this can be converted to the following probability statement: With probability at least $1 - 2^{-53}$ there exist 2^{53} polynomials of degree at most 15 that agree with the given data points on two points each in each of the 10 categories. (Thus, we achieve roughly a 53-bit security level.)

Finally, we give a general characterization of the information-theoretic security of Lock according to Definition 2. A proof, again, may be found in the appendix.

Theorem 6. *For every $\delta > 0$, the algorithm* Lock *is (δ, p)-information theoretically secure for $p = 2\sqrt{\frac{1}{3} q^{k-(1+\delta)t} (r/t)^{(1-\delta)t}}$*

Remark: It is possible to make a substantially stronger security claim under reasonable computational assumptions on the hardness of Reed-Solomon decoding, as done in, e.g., [18]. We do not explore this possibility here, as it is not essential to our achieving good security results, and there is no general consensus about appropriate hardness assumptions for this problem.

5 Implementation Issues

In the examples we have given above, the security parameterization is slightly weak from a cryptographic standpoint. For instance, in example 4, we achieve roughly a 53-bit security level, slightly weaker than DES, and thus vulnerable to intensive off-line attack [11]. In many cases, we may address this problem simply by re-parameterizing the vault, in particular, raising the size t of the set A. For example, to construct a strong personal-entropy system, we might require the user to answer 29 questions correctly out of 32 (with $t = 32$ and $k = 25$), thereby achieving a fuzzy vault with 85-bit security, which offers good cryptographic security for most purposes.

For applications involving privacy-protected matching, such as the movie-lover's problem, this approach will not work. The problem here lies not in the fuzzy vault scheme, but is inherent in the problem of matching people or other entities. In particular, given that there are only six billion or so people in the world, if Alice wants to have a good chance of finding a compatible movie lover, she can at best select a set that can be guessed by a well-informed attacker with a probability of one in six billion or so. As six billion is equal roughly to $2^{32.3}$, this means that Alice's vault can at best be expected to have less than 33-bit security[6] – entirely inadequate for cryptographic purposes. In most scenarios involving matching, the pool of participants, and thus the best achievable off-line security, is likely to be even smaller.

For this reason, we propose that fuzzy vaults in such privacy-protecting matching scenarios are best employed in an on-line setting. Here is a rough, first sketch of an idea for an on-line version of the movie-lover's problem. Alice publishes a fuzzy vault V_A encrypting a secret value κ under a set A of her favorite movies. If Bob wishes to obtain Alice's telephone number, he tries to open V_A using his set of favorite movies B. If he decodes successfully, he obtains a secret value κ'. Alice and Bob now invoke a *password-authenticated key-agreement protocol* (see [5] for a recent example). They use their respective secrets κ and κ' as passwords for this protocol.[7] If $\kappa = \kappa'$, then Alice and Bob will successfully establish a private channel. Otherwise, they will learn no information about the secret value of the other party, aside from the fact that $\kappa \neq \kappa'$. By employing an appropriate throttling mechanism, Alice can restrict the number of overall queries or queries from a single source so as to restrict information leakage.

This strategy effectively protects Alice's set A, but enables Alice to attract inappropriate matches. To illustrate this problem, consider the following simple strategy. Alice creates a vault V_A in which *all* x-coordinates agree with some polynomial p. Now, no matter what Bob inserts into his set B, he will think that Alice has the same favorite movies. This example highlights the important fact that a fuzzy vault is not, strictly speaking a commitment. In particular, it is not uniquely *binding*: Alice may embed multiple sets $A_1, A_2, \ldots, A_l$ in a single vault V_A.

There are several ways to avoid this difficulty. One way is for Alice to include in her vault a cryptographically binding commitment c_A to her secret value κ using, such as, e.g., a Pedersen commitment [19]. Now, Alice participates in the key agreement protocol with Bob in a manner that binds her to c_A (through a straightforward modification of existing algorithms). This does not ensure that V_A and c_A represent the same secret κ, but this condition is not required for secure matching. A more general, mutually binding protocol is the following.

1. Alice publishes V_A on secret κ_A.
2. Bob publishes V_B on secret κ_B.
3. Alice applies UNLOCK to V_B using her set A. If successful, she obtains a value κ'_B. Otherwise, she aborts.
4. Bob applies UNLOCK to V_A using his set B. If successful, he obtains a value κ'_A. Otherwise, he aborts.
5. Alice and Bob do a password-authenticated key agreement using respective passwords $(\kappa_A \parallel \kappa'_B)$ and $(\kappa_B \parallel \kappa'_A)$. (Here, $\parallel$ is some suitable conjunctive operator.)

Due to lack of space, we do not undertake a formal analysis of this protocol, but merely note that it binds Alice and Bob to use of their respective vaults V_A and V_B. Given use of a zero-knowledge password-authenticated key-agreement algorithm, the above protocol reveals to Alice and Bob only the information contained in V_A and V_B and also whether $\kappa_A = \kappa_B$. We reserve defer of such protocols for further research.

[6] It is possible to slow the algorithm UNLOCK so as to impose a higher computational burden on an attacker, but this approach still doesn't offer adequate security here.

[7] Alternatively, as a practical alternative more compatible with existing infrastructure, they can employ the Secure Socket-Layer (SSL) protocol to establish a private, authenticated channel, and then employ a *socialet millionaires'* or similar protocol to test the condition $\kappa = \kappa'$ in zero knowledge [14, 21]. This method depends upon one player having an appropriately signed certificate.

Acknowledgments

The authors wish to extend special thanks to Burt Kaliski for his ideas regarding Section 5. Thanks also to Daniel Bleichenbacher, Markus Jakobsson, and the anonymous reviewers of this paper on the Eurocrypt 2000 program committee for their many helpful 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, *Asiacrypt '94*, pages 238–248. Springer-Verlag, 1994. LNCS No. 917.
2. C.H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. Practical quantum oblivious transfer protocols. In J. Feigenbaum, editor, *Crypto '91*, pages 351–366. Springer-Verlag, 1991. LNCS No. 576.
3. E. R. Berlekamp. *Algebraic Coding Theory.* McGraw Hill, New York, 1968.
4. D. Bleichenbacher and P. Nyuyen. Noisy polynomial interpolation and noisy chinese remaindering. In B. Preneel, editor, *Eurocrypt '00*, pages 53–69, 2000. LNCS no. 1807.
5. V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using diffie-hellman. In B. Preneel, editor, *Eurocrypt '00*, pages 156–171. Springer-Verlag, 2000. LNCS no. 1807.
6. 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.
7. 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.
8. 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.
9. I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 475–484, 1999.
10. C. Ellison, C. Hall, R. Milbert, and B. Schneier. *Protecting Secret Keys with Personal Entropy*, pages 311–318. 2000.
11. Electronic Frontier Foundation. Cracking DES: Secrets of encryption research, wiretap politics & chip design. O'Reilly, 1998.
12. V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *FOCS '98*, pages 28–39. IEEE Computer Society, 1998.
13. T. Jakobsen. Cryptanalysis of block ciphers with probabisetic non-linear relations of low degree. In H. Krawczyk, editor, *Crypto '98*, pages 212–222. Springer-Verlag, 1998. LNCS No. 1462.
14. M. Jakobsson and M. Yung. Proving with knowing: On oblivious, agnostic, and blindfolded provers. In N. Koblitz, editor, *Crypto '96*, pages 186–200. Springer-Verlag, 1996. LNCS no. 1109.
15. A. Juels and M. Wattenberg. A fuzzy commitment scheme. In G. Tsudik, editor, *Sixth ACM Conference on Computer and Communications Security*, pages 28–36. ACM Press, 1999.
16. J. L. Massey. Shift register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.
17. R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical Report DSN progress report 42-44, Jet Propulsion Laboratory, Pasadena, 1978.
18. F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In G. Tsudik, editor, *Sixth ACM Conference on Computer and Communications Security*, pages 73-82. ACM Press, 1999.
19. T. Pedersen Non-interactive and information-theoretic secure verifiable secret sharing In J. Feigenbaum, editor, *Crypto '91*, pages 129–140. Springer-Verlag, 1991. LNCS No. 576.
20. W. W. Peterson. Encoding and error-correction procedures for Bose-Chaudhuri codes. *IRE Transactions on Information Theory*, IT-60:459 – 470, 1960.
21. B. Schoenmakers, F. Boudot, and J .Traoré. A fair and efficient solution to the sociaset millionaires' problem. *Discrete Applied Mathematics*, 2000. To appear.

22. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
23. C. Soutar. Biometric encryption for secure key generation, January 1998. Presentation at the 1998 RSA Data Security Conference.
24. 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.
25. J. Stern. A new identification scheme based on syndrome decoding. In D.R. Stinson, editor, *Crypto '93*, pages 13–21. Springer-Verlag, 1993. LNCS No. 773.

A Proofs

Lemma 4. *For every $\mu > 0$, with probability at least $1 - \mu$, the target set R generated by the algorithm* Lock *on polynomial p and locking set A satisfies the following condition: There exist at least $\frac{\mu}{3} q^{k-t}(r/t)^t$ polynomials $p' \in \mathcal{P}$ such that R includes at least t points of the form $(x, p'(x)) \in \mathcal{F} \times \mathcal{F}$.*

Proof: We first analyze a modified algorithm whose output is just a collection of r random points, say (x_i, z_i) in $\mathcal{F} \times \mathcal{F}$. We claim that the expected number of polynomials that agree with this set of points in t places, denoted N, is large. To see this, fix a polynomial p' and consider the probability that p agrees with the set of points in exactly t places. This probability is given by the expression:

$$\binom{n}{t} q^{-t}(1 - 1/q)^{r-t} \geq \frac{1}{3}\left(\frac{r}{qt}\right)^t.$$

Thus the expected number of polynomials of degree less than k that agree with t of the r random points is $N = \frac{1}{3} q^k \left(\frac{r}{qt}\right)^t$.

We now revert to our original problem, where the algorithm Lock is not outputting a random set of points, but rather a set of points t of which are selected to be from a fixed polynomial p. We will show that in this case the probability that the number of polynomials in agreement with the output set in t points is than μN, is at most μ. This yields the lemma as desired. To prove this part, we construct a (huge) bipartite graph $\mathcal{G}$. The left vertices correspond to polynomials of degree less than k (i.e., there is one vertex for each such polynomial). The right vertices correspond to subsets of n points from $\mathcal{F} \times \mathcal{F}$ (i.e., there is one vertex for every subset of size r). Two vertices p and R are adjacent if p agrees with the set in exactly r points. The analysis of the modification of Lock showed that a average degree of a right vertex is N. Notice that the algorithm Lock, on the other hand, picks a fixed vertex p on the right and outputs a random neighbor of this vertex. Our goal is to show that this output vertex has high degree (say μN) with high probability. Towards this end, we first notice that the graph $\mathcal{G}$ is symmetric about left vertices, i.e., for every pair of vertices on the left p and p' there is an automorphism of $\mathcal{G}$ that maps p to p'. Thus it would suffice to consider the right endpoint of a random edge of $\mathcal{G}$ and argue that its degree is at least μN. But this is obviously true since vertices with degree $< \mu N$ can only account for a μ fraction of the edges in the graph $\mathcal{G}$. The lemma is thus proved. □

Lemma 5. *For every $\mu > 0$, with probability at least $1-\mu$, the target set R generated by the algorithm* Lock *to commit to a polynomial p with locking set A satisfies the following condition: There exist at least $\frac{\mu}{3} q^{k-t}|\mathcal{E}|$ polynomials $p' \in \mathcal{P}$ such that R agrees with p' on some subset of points from the family $\mathcal{E}$.*

Proof: This lemma is proved in exactly the same way as the previous one. The only change is that the probability that a given polynomial p' agrees with a random set of r points in one of the subsets from $\mathcal{E}$ reduces to

$$|\mathcal{E}|(1/q)^t(1 - 1/q)^{r-t},$$

and this change percolates through the rest of the proof. $\square$

Theorem 6. *For every* $\delta > 0$ *the algorithm* LOCK *is* (δ, μ)*-information theoretically secure for* $\mu = 2\sqrt{\frac{1}{3}q^{k-(1+\delta)t}(r/t)^{(1-\delta)t}}$.

Proof: As in the proofs of Lemma 4 and Lemma 5, we first argue that given any $\mu > 0$, $E \subseteq \mathcal{U}$ of size t and $E' \subseteq E$ of size δt, the data points R have at least $\frac{\mu}{3}q^{k-(1+\delta)t}(r/t)^{(1-\delta)t}$ polynomials agreeing with the data points on t points that include E' as a subset.

Picking $\mu = \sqrt{\frac{1}{3}q^{k-(1+\delta)t}(r/t)^{(1-\delta)t}}$ we get that with probability at least $1 - \mu$, there are at least $1/\mu$ polynomials exhibiting such behavior. By symmetry, each of these polynomials is equally likely to be the polynomial p that is being encrypted. Thus the Attack algorithm has at most a 2μ chance of finding p — success probability of μ, if our encryption is unlucky and there are not too many polynomials agreeing with the data; and μ is the probability that the encryption is correct, but the attacker manages to guess it by luck. Thus we get the security as claimed. $\square$

This article was processed using the LaTeX macro package with LLNCS style