Fuzzy Vault for Multiple Users

No Author Given

No Institute Given

Abstract. We introduce an extension of the Fuzzy Vault scheme to support multiple users. Namely, different participants might share the same vault to store their secret. In the classical Fuzzy Vault, a user locks a secret using a set A and one can unlock it by providing another set B which overlaps substantially with A. In our extension, our Extented Fuzzy Vault is locked thanks to different sets A^j, $j = 1, \ldots, l$ and can be unlocked with a set B with enough common elements with one of these A^j. Our results are based on Folded Reed-Solomon Codes and their recent list recovery algorithm. This way, our idea can be interpreted as a natural extension of the list decoding of the Reed-Solomon Codes to list recovery. We give a security analysis of our proposal relying on Secure Sketches security properties to gauge our results. Finally, we provide details on an implementation to support our ideas.

Keywords. Fuzzy Vault, Folded Reed-Solomon Codes, list recovery algorithm

1 Introduction

Since its introduction in 2002 by Juels and Sudan [7,8], the Fuzzy Vault (FV) scheme has attracted a lot of attention; in particular in the biometric community [2,9,16,13,1,12,10,11,17,19,18,23]. This Fuzzy Vault scheme relies on the Reed-Solomon codes and their decoding capacity. More precisely, to contrust a FV scheme, each information symbol x_i – considered as element of a finite field – is accompanied by its evaluation by a given polynomial p to form a pair $(x_i, y_i = p(x_i))$. Moreover, some chaff points are added to thwart an adversary who should not be able to differentiate genuine pairs from fake ones. Whenever a user comes back to recover data from a FV, he has to produce a sufficient number of genuine pairs; this way, the errors decoding capacity of the Reed-Solomon code enables to recover the underlying polynomial p.

In this paper, we introduce the idea of an Extended Fuzzy Vault (EFV) scheme for multiple users. This means that following the idea of FV, different users can now share the same EFV. From a technical point of view, an immediate difficulty comes from the fact that different users can have the same information symbol x_i. To deal with this problem, we replace the underlying list decoding of Reed-Solomon codes suggested by FV scheme [8] by a more advanced technique called list recovery. Indeed, this is the needed method to handle multiplicities we can now encounter for each x_i. Doing so, we also switch from Reed-Solomon

codes to a variant called folded Reed-Solomon codes. Folded Reed-Solomon codes are an explicit family of codes that achieve the optimal trade-off between rate and error-correction capability. Moreover, recently [4] introduces effective list decoding algorithms for this family of codes which can be easly extented to list recovery algorithms [5].

The origin of the motivation for our extension can be found in [8] where the case of encoding multiple sets in the same fuzzy vault is briefly mentioned. For instance, the idea of replacing part of, or all, chaff points by points corresponding to other polynomials is introduced. However, [8] does not try to deal with potential collisions between the different sets to be encoded.

On an applicative side, and in particular in the biometric domain, the demand of encoding multiple sets is natural [15]. Today, the FV scheme permits to users to authenticate themselves through their biometric data. For an identification scenario, where a comparison with a database of several users is needed, [15] studies the way to reduce the number of comparisons to run among the several vaults – each corresponding only to one set – by filtering techniques. Going further, our proposal can directly address identification protocols, which is the largest application of biometric recognition process, by seeing the whole database as a single vault.

2 Fuzzy Vault (FV) Scheme

Let $\mathcal{F}$ stand for a finite field of cardinality q ($\mathcal{F} = GF(q)$). We have a secret value $\kappa \in \mathcal{F}^k$ to be protected and a secret set $A = \{a_i \in \mathcal{F}, i = 1, \ldots, t\}$. Let r, a parameter of the scheme, be an integer greater than t.

The associated Fuzzy Vault (FV) $V_A \in \mathcal{F}^{2r}$, as introduced by Juels and Sudan in [7,8], is computed thanks to the LOCK algorithm that takes as inputs the set A and a vector κ. This algorithm transforms κ into a polynomial p. Moreover, to each value in A, its evaluation by the polynomial p is made to form a pair. Finally, some chaff points, i.e. points taken at random are chosen. This gives:

LOCK$(A, \kappa) = V_A = \{(x_i, y_i) \in \mathcal{F} \times \mathcal{F}, i = 1, \ldots, r\}$ with

for $i = 1$ **to** t **do**
  $x_i = a_i$, $y_i = p(x_i)$
end
for $i = t + 1$ **to** r **do**
  $x_i \in \mathcal{F} \setminus A$, $y_i \in \mathcal{F} \setminus \{p(x_i)\}$ (both taken randomly)
end

Whenever one gets a set B with many common elements with the secret set A, he should be able to recover the secret κ from V_A. To this end, an algorithm UNLOCK is provided. This gives:

Definition 1 ([8]) *A locking/unlocking pair (*LOCK*;* UNLOCK*) with parameter set $(k;t;r)$ is complete with ϵ-fuzziness if the following holds. For $\kappa \in \mathcal{F}^k$, for 2 sets A, $B \in \mathcal{F}^t$ such that $\|A-B\| \leq \epsilon$, we have* $\mathrm{UNLOCK}(B;\mathrm{LOCK}(A;\kappa)) = \kappa$ *with overwhelming probability.*

In [8], the UNLOCK algorithm relies on the error correction capacity of the Reed-Solomon codes.

A Reed-Solomon code over the finite field $\mathcal{F}$, of length t, and dimension k, is made by the evaluations of polynomials of $\mathcal{F}[X]$ of degree strictly less than k for t elements of $\mathcal{F}$. Each polynomial p is associated with the codeword $(p(a_1),\ldots,p(a_t))$ where the a_i's stand for elements in $\mathcal{F}$. Reed-Solomon codes can be decoded up to $\frac{t-k}{2}$ errors by the Peterson-Berlekamp-Massey algorithm [14]. List decoding algorithms [22], such as the one introduced by Guruswami and Sudan in [6], enable to correct even more errors. We denote by RSDECODE such a decoding algorithm which takes as input a set of t points of $\mathcal{F}^2$ and returns either a polynomial of degree k or null.

To achieve complete $\frac{t-k}{2}$-fuzziness for the (LOCK; UNLOCK) pair with $k \leq t \leq r \leq q$, one has to simply take in the Fuzzy-Vault V_A, the points with an abscissa in B and recover the underlying polynomial using the RSDECODE Peterson-Berlekamp-Massey decoding algorithm.

In [8] the security analysis is formalized through an attack model that captures the possibility for an attacker to have prior knowledge of a part of the set A when trying to find out the value of κ from $\mathrm{LOCK}(A;\kappa)$: Given an uniformly random subset A' of A of a given bounded size, the probability for the adversary to guess the right value κ (given A' and $\mathrm{LOCK}(A;\kappa)$) should remain small.

The security is directly depending on the number $r-t$ of chaff points that are added. In fact, the more chaff points they are, the more possible polynomials of degree less than k are encoded in the vault, thus leading to higher conditional entropy of κ knowing the vault V_A. The authors quantify this by showing that for any $\mu > 0$ there is, with probability at least $1-\mu$, at least $\frac{\mu}{3}q^{k-t}(\frac{r}{t})^t$ possible polynomials. This enables to prove the security for uniformly drawn set A. The case of non-uniform distribution is also analyzed when $r = q-1$.

Another security model to analyze the Fuzzy Vault scheme is to see it in the secure sketch [3] framework. Let $\mathcal{H}$ be a metric space with distance function d. The formal definition of secure sketches functions is the following.

Definition 2 *A $(\mathcal{H}, m, m', t)$-secure sketch is a pair of functions $(\mathsf{SS}, \mathsf{Rec})$ where the sketching function SS takes $w \in \mathcal{H}$ as input, and outputs a sketch in $\{0,1\}^*$, such that for all random variables W over $\mathcal{H}$ with min-entropy $\mathbf{H}_\infty(W) \geq m$, we have the conditional min-entropy $\overline{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \geq m'$.*

The recovery function Rec takes a sketch P and a vector $w' \in \mathcal{H}$ as inputs, and outputs a word $w'' \in \mathcal{H}$, such that for any $P = \mathsf{SS}(w)$ and $\mathrm{d}(w,w') \leq t$, it holds that $w'' = w$.

*Where $\mathbf{H}_\infty(X) = -\log_2 \max_x \mathbb{P}(X = x)$ stands for the **min-entropy** and $\overline{\mathbf{H}}_\infty(X \mid Y) = -\log_2 \mathbf{E}_{y \leftarrow Y}(2^{-\mathbf{H}_\infty(X|Y=y)})$ for the **conditional min-entropy**.*

The difference between m and m' gives the entropy loss of the scheme. In [21], it is shown that the entropy loss of the fuzzy vault scheme is at most

$$(t-k)\log_2 q + \log_2 \binom{q}{r} - \log_2 \binom{q-t}{r-t}.$$

This leads to the lower bound

$$\log_2 \binom{r}{t} - (t-k)\log_2 q \tag{1}$$

on the remaining entropy $\overline{\mathbf{H}}_\infty(A \mid \mathrm{LOCK}(A;\kappa))$.

One advantage of this measure compared to the original security analysis from [7,8] is that it is independent on the distribution assumption of the encoded set A (in particular this handles non-uniformity for all $r \leq q-1$).

Example 1 *The movie lover's problem is introduced in [8] as an example of application of the FV scheme. In this problem, Alice – a movie lover – has to choose among* 10^4 *movies her 22 favorites. She is looking for someone with similar affinities. She does not want to reveal her preferences to other people. She uses the FV scheme to encrypt her phone number as whenever someone unlocks her vault, he demonstrates that he shares her film taste and this enables him to call her.*

3 Folded Reed-Solomon Codes

After recalling some definitions from [5], we present the list decoding algorithm taken from [4] used in our scheme.

3.1 A few definitions

Definition 3 (Folded Reed-Solomon Code) *Given γ a generator of $\mathcal{F}$, the m-folded version of the Reed Solomon code $C[n,k]$, denoted $FRS_{\mathcal{F},\gamma,m,N,k}$, is a code of block length $N = n/m$ over $\mathcal{F}^m$ where $n = q-1$ is divisible by m. The encoding of a message $p \in \mathcal{F}[X]$ of degree at most $k-1$ is given by*

$$p(X) = \left(\begin{bmatrix} p(1) \\ p(\gamma) \\ \vdots \\ p(\gamma^{m-1}) \end{bmatrix}, \begin{bmatrix} p(\gamma^m) \\ p(\gamma^{m+1}) \\ \vdots \\ p(\gamma^{2m-1}) \end{bmatrix}, \ldots, \begin{bmatrix} p(\gamma^{n-m}) \\ p(\gamma^{n-m+1}) \\ \vdots \\ p(\gamma^{n-1}) \end{bmatrix} \right)$$

The m-tuple $(p(\gamma^{jm}), p(\gamma^{jm+1}), \ldots, p(\gamma^{jm+m-1}))$ is the j'th symbol of the encoding of p for $0 \leq j < n/m$.

More generally any restriction of $p(X)$ on part of its columns will form a codeword of a folded Reed Solomon code $FRS_{\mathcal{F},\gamma,m,N,k}$ of shorter length $N = n/m$ with $n < q-1$. In the paper, we consider this general case where n can be either equal to $q-1$ or lower than $q-1$.

Definition 4 (List decodable code) *Let C be a code of block length N, let $\sigma \geq 1$ be an integer and $0 < \rho < 1$ be a real. C is called (ρ, σ)-list decodable if every Hamming ball of radius ρN has at most σ codewords in it.*

List decoding is a relaxation of unique decoding. Instead of outputting a single codeword, given an error bound e, the decoding algorithm gives the list of all polynomials $p \in \mathcal{F}[X]$ of degree at most $k-1$ whose encoding differs with the received word in at most e symbols.

List recovery is an extension of list decoding where for each position i of the received message the input is of the form of a set T_i of possible values.

Definition 5 (List recoverable code) *Let C be a code of block length N, let $\ell, \sigma \geq 1$ be integers and $0 < \rho < 1$ be a real. C is called (ρ, ℓ, σ)-list recoverable if for all sequences of sets $T_0, \ldots, T_{N-1}$ where $|T_i| \leq \ell$, for every $0 \leq i \leq N-1$, there are at most σ codewords $\mathbf{c} = \langle c_0, \ldots, c_{N-1} \rangle$ such that $c_i \in T_i$ for at least $(1-\rho)N$ positions i.*

For $\ell = 1$, a $(\rho, 1, \sigma)$-list recoverable code C is (ρ, σ)-list decodable.

3.2 List decoding of folded Reed-Solomon codes

We chose to use Guruswami's list decoding algorithm [4] from which we give here a brief overview. Decoding of FRS codes is a two-step process: first interpolate a multivariate polynomial $Q \in \mathcal{F}[X, Y_1, \ldots, Y_s]$, where s is a chosen a parameter lower than m, and secondly find all candidate polynomials p of degree $k-1$ satisfying $Q(X, p(X), \ldots, p(\gamma^{s-1}X)) = 0$. Both steps can be reduced to the solving of linear systems over $\mathcal{F}$.

Interpolation Starting from a received word $\mathbf{y} \in (\mathcal{F}^m)^N$ of the form $\langle \mathrm{y}_0, \ldots, \mathrm{y}_N \rangle = \langle [y_0, \ldots, y_{m-1}], \ldots, [y_{n-m}, \ldots, y_{n-1}] \rangle$, we are looking for a nonzero polynomial

$$Q(X, Y_1, Y_2, \ldots, Y_s) = A_0(X) + A_1(X)Y_1 + A_2(X)Y_2 + \ldots + A_s(X)Y_s \quad (2)$$

over $\mathcal{F}$ where $deg(A_i) \leq D$ for $i = 1, \ldots, s$ and $deg(A_0) \leq D + k - 1$ and D is chosen as

$$D = \left\lfloor \frac{N(m-s+1) - k + 1}{s+1} \right\rfloor \quad (3)$$

For $i = 0, .., N-1, j = 0, .., m-s$, Q has to satisfy

$$Q(\gamma^{im+j}, y_{im+j}, y_{im+j+1}, \ldots, y_{im+j+s-1}) = 0 \quad (4)$$

The number $N(m-s+1)$ of these interpolation conditions are smaller than the number $(D+1)s + D + k$ of monomials in Q, which guarantees the existence of such a polynomial. Note that we can choose *any* polynomial Q satisfying the conditions (4).

From the choice of D (3), we can deduce that the fractional agreement needed to decode a received word $\mathbf{y}$ is $\tau > \frac{1}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1}$ where $R = k/n$ is the rate of the code. This is equivalent to say that we can tolerate up to $e = \frac{s}{s+1}(N - \frac{k}{m-s+1})$ erroneous symbols in $\mathbf{y}$.

Root finding We need now to find all polynomials $p \in \mathcal{F}[X]$ satisfying

$$A_0(X) + A_1(X)p(X) + A_2(X)p(\gamma X) + \ldots + A_s(X)p(\gamma^{s-1}X) = 0 \qquad (5)$$

This equation can be seen as a linear system in the coefficients $p_0, \ldots, p_{k-1}$ of p. Let us define $B(X) = a_{1,0} + a_{2,0}X + \ldots + a_{s,0}X^{s-1}$ and we denote $A_i(X) = \sum_{j=0}^{D+k-1} a_{i,j}X^j$ for $0 \leq i \leq s$. Then for $r = 0, \ldots, k-1$ we can deduce from equation (5) that

$$\begin{aligned} &a_{0,r} + p_r(a_{1,0} + a_{2,0}\gamma^r + \ldots + a_{s,0}\gamma^{(s-1)r}) + p_{r-1}(a_{1,1} + a_{2,1}\gamma^{r-1} + \ldots + a_{s,1}\gamma^{(s-1)(r-1)}) \\ &+ \ldots + p_1(a_{1,r-1} + a_{2,r-1}\gamma + \ldots + a_{s,r-1}\gamma^{s-1}) + p_0(a_{1,r} + \ldots + a_{s,r}) = 0 \qquad (6) \end{aligned}$$

which is equivalent to

$$B(\gamma^r)p_r + (\sum_{i=0}^{r-1} b_i^{(r)} p_i) + a_{0,r} = 0 \qquad (7)$$

This leads to a lower-triangular linear system with on its diagonal the $B(\gamma^r)$'s. If $B(\gamma^r) \neq 0$, p_r is an affine combination of $p_0, \ldots, p_{r-1}$. The dimension of the space of solutions is thus limited by the amount of $B(\gamma^r) = 0$. As γ is a generator of $\mathcal{F}$, all γ^r values for $0 \leq r < k$ are different. Moreover B is a $s-1$ degree polynomial, thus $B(\gamma^r)$ vanishes for at most $s-1$ values of r.

The whole decoding procedure can be summed up as follows

Theorem 1 ([4]) *Given a received word $\mathbf{y} \in (\mathcal{F}^m)^N$, for any integer s, $1 \leq s \leq m$, one can find a subspace of dimension at most $s-1$ containing all polynomials $p \in \mathcal{F}[X]$ of degree less than k whose FRS encoding agrees with $\mathbf{y}$ in at least a fraction $\frac{1}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1}$ of the N codeword symbols.*

According to Definition 4, for any integer $1 \leq s \leq m$, an $FRS_{\mathcal{F},\gamma,m,N,k}$ code is an (ρ, q^{s-1})-list decodable code with $\rho = 1 - (\frac{1}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1})$. The algorithm described previously gives the corresponding list decoding procedure.

Remark 1 *The fuzzy vault construction from Section 2 can be adapted to the folded Reed-Solomon codes. The y-axis points of the vault will still be associated to one coordinate of a codeword associated to a random polynomial; but with a coordinate that corresponds here to several evaluations (one column of $p(X)$ in Definition 3). Based on the list decoding property – here applied with a folded Reed-Solomon code of length t with t the size of the locked set as in Section 2 – this would lead to a fuzzy vault scheme ensuring $\frac{s}{s+1}(t - \frac{k}{m-s+1})$-fuzziness.*

3.3 Generalization to list recovery

Suppose now that for each position $i = 0, \ldots, N-1$ of a received message we have a set T_i of possible values $\mathrm{y}_{i,1}, \ldots, \mathrm{y}_{i,\#T_i}$ with $\mathrm{y}_{i,j} = [y_{im,j}, \ldots, y_{im+m-1,j}] \in \mathcal{F}^m$ and $1 \leq j \leq \#T_i \leq \ell$. Fortunately, list recovering is also possible with Guruswami's algorithm [4] but with some slight modifications. As we have now up to ℓ values for each symbol, we need to expand the $N(m-s+1)$ interpolation conditions to $\ell N(m-s+1)$ to take all symbols into account. For $i = 0, .., N-1, j = 0, .., m-s$, Q has now to satisfy

$$\begin{gathered} Q(\gamma^{im+j}, y_{im+j,1}, y_{im+j+1,1}, \ldots, y_{im+j+s-1,1}) = 0 \\ Q(\gamma^{im+j}, y_{im+j,2}, y_{im+j+1,2}, \ldots, y_{im+j+s-1,2}) = 0 \\ \vdots \\ Q(\gamma^{im+j}, y_{im+j,\#T_i}, y_{im+j+1,\#T_i}, \ldots, y_{im+j+s-1,\#T_i}) = 0 \end{gathered}$$

The same way, D becomes

$$D = \left\lfloor \frac{\ell N(m-s+1) - k + 1}{s+1} \right\rfloor \tag{8}$$

The interpolation step works just as before but with a bigger linear system to solve and root finding is exactly the same. Finally, we have the new agreement fraction $\tau > \frac{\ell}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1}$.

This means that an $FRS_{\mathcal{F},\gamma,m,N,k}$ code is an (ρ, ℓ, q^{s-1})-list recoverable code with $\rho = 1 - (\frac{\ell}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1})$.

Remark 2 *Following the preceding facts, one has to note that there are some restrictions on the values of parameters m, s and ℓ, indeed we need to have $1 \leq \ell \leq s \leq m$.*

4 Extended Fuzzy Vault (EFV) Scheme

In this section, we show how to extend the FV scheme to the case where the vault is made thanks to different secret sets A^j, $j = 1, \ldots, l$. As before for the FV scheme, we want to unlock the vault from a set B which contains enough common elements with one of these A^j's.

Let $\mathcal{E}$ be an alphabet containing N symbols denoted by $x_1, \ldots, x_N$. Each A^j possesses t of these symbols.

We will rely on the list recovery capacity of the Folded Reed-Solomon Codes as described in the previous section to build our EFV scheme. We take back the notations of Section 3, $N = n/m$ is the length of the underlying Folded Reed-Solomon Code in the finite field $\mathcal{F}$. As before γ is a generator of its multiplicative group. Let $F_1, \ldots, F_l$ be l evaluation functions according to Definition 3. More precisely, each set A^j is associated to a function F_j. Let p_j a random polynomial of degree $k-1$ in $\mathcal{F}[X]$. The value $F_j(x_i)$ is defined as the i'th symbol of the encoding of p_j for $0 \leq i < N$, $(p_j(\gamma^{im}), p_j(\gamma^{im+1}), \ldots, p_j(\gamma^{im+m-1}))$.

4.1 LOCK

We take back the idea of the Fuzzy Vault scheme that is made with genuine polynomial evaluations hidden within a cloud of chaff points.

The extended vault $V = \text{LOCK}(A^1, \ldots, A^l; \kappa)$ corresponding to the secret sets A^j is made as follows (κ represents here the randomness used to generate the l evaluation functions $F_1, \ldots, F_l$). Consider N sets $S_1, \ldots, S_N$ that will be related to the N symbols in $\mathcal{E}$. Let r, ℓ be two integers corresponding to parameters for the addition of chaff points.

Initialize a counter $cpt \leftarrow 0$ and the sets $S_1, \ldots, S_N$ to the empty set:
$S_i \leftarrow \emptyset$, $i = 1, \ldots, N$
for $i = 1$ **to** N **do**
  for $j = 1$ **to** l **do**
    if $x_i \in A^j$ **then**
      $S_i \leftarrow S_i \cup \{F_j(x_i)\}$
    end
  end
  set $l_i \leftarrow \#S_i$
  if $l_i \neq 0$ **then**
    $cpt \leftarrow cpt + 1$
    for $j = l_i + 1$ **to** ℓ **do**
      let y_i^j randomly chosen in $\mathcal{F}^m \setminus \{F_d(x_i)\}_{d=1,\ldots,l}$
      let $S_i \leftarrow S_i \cup \{y_i^j\}$
    end
  end
end
randomly choose $i_{cpt+1}, \ldots, i_r$ such that $\#S_{i_e} = 0$ for $e \in \{cpt + 1, \ldots, r\}$
for $e = cpt + 1$ **to** r **do**
  for $j = 1$ **to** ℓ **do**
    let $y_{i_e}^j$ randomly chosen in $\mathcal{F}^m \setminus \{F_d(x_{i_e})\}_{d=1,\ldots,l}$
    let $S_{i_e} \leftarrow S_{i_e} \cup \{y_{i_e}^j\}$
  end
end
Finally, every set S_i for $i = 1, \ldots, N$ is shuffled

At the end, the locked vault is $V = \text{LOCK}(A^1, \ldots, A^l) = \{(x_i, S_i), i = 1, \ldots, N\}$.

To simplify the description and the security analysis, the same number of points is associated to each set S_i. Note that this constraint could be relaxed.

Remark 3 *In other words, we construct a collection of sets $S_1, \ldots, S_N$ where*

- *there are $N - r$ empty sets*

– *and r indices $i_1, \ldots, i_r$ such that for $e \in \{1, \ldots, r\}$, S_{i_e} is constituted with the union of the sets*

$$S_{i_e}^{(1)} = \{F_j(x_{i_e}) | x_{i_e} \in A^j,\ j = 1, \ldots, l\}$$

with some chaff points

$$S_{i_e}^{(2)} = \{y_{i_e}^j \in \mathcal{F}^m \setminus \{F_d(x_{i_e})\}_{d=1,\ldots,l} \,|\, j = 1, \ldots, \rho\}$$

such that $\#S_{i_e}^{(1)} + \#S_{i_e}^{(2)} = \ell$.

As for the original fuzzy vault scheme, we have a number of chaff points that hide the genuine positions that correspond to polynomial values. Moreover we now have another dimension where at one given position, a genuine polynomial value is mixed among other polynomial values and additional chaff points. And even the genuine sets (the A^j's) can be seen as chaff with respect to one given set A^i.

4.2 UNLOCK

Let $A^1, \ldots, A^l$ be l sets of size t. The unlocking procedure of a vault

$$\text{LOCK}(A^1, \ldots, A^l) = \{(x_i, S_i), i = 1, \ldots, N\}$$

is as follows.

When receiving a new set $B = \{b_1, \ldots, b_t\}$, take the subcollection $S_{i_1}, \ldots, S_{i_t}$ where the i_e's are such that $x_{i_e} = b_e$ and run the list recovery decoding algorithm with input $((x_{i_1}, S_{i_1}), \ldots, (x_{i_t}, S_{i_t}))$. Note that here the list recovery decoding algorithms is in fact executed by restricting the folded RS code to a folded RS code $FRS_{\mathcal{F},\gamma,m,t,k}$ of length t.

It will output the evaluation functions F_j corresponding to the A^j's that are sufficiently close to B. From the F_j's, this leads to retrieving the content of A^j. More precisely, based on the parameters of the list recovery decoding algorithm, for any s such that $\ell \leq s \leq m$, we can retrieve the A^j's such that $\|A^j - B\| \leq \epsilon$ with $\epsilon = \frac{1}{s+1}((s+1-\ell)t - \frac{sk}{m-s+1})$.

Lemma 1. *Following a straightforward extension of Definition 1, our Extended Fuzzy Vault scheme based on a $FRS_{\mathcal{F},\gamma,m,N,k}$ code is complete with ϵ-fuzziness where*

$$\epsilon = \frac{1}{s+1}((s+1-\ell)t - \frac{sk}{m-s+1})$$

Remark 4 *Although our EFV construction is described in the folded Reed-Solomon context, it can also be applied to other codes for which a list recovery algorithm exists. This is for instance the case for classical Reed-Solomon codes. So the EFV scheme can be instantiated for RS codes but then the list recovery decoding will be suboptimal [20].*

5 Security Properties of the EFV scheme

As in [8], we can estimate the difficulty for an attacker to retrieve a genuine polynomial among the different possible combinations of points in the vault by approximating the number of polynomials. As explained in Section 2, for the Reed-Solomon fuzzy vault of dimension k over $\mathcal{F}$ with t genuine points and $r-t$ chaff points, for any $\mu > 0$ there is at least $\frac{\mu}{3} q^{k-t} (\frac{r}{t})^t$ possible polynomials, with probability at least $1-\mu$.

In the case of the application of the EFV scheme to RS codes, following the fact that one can extract ℓ^r different possible combinations of r pairs (x_i, y_i) (with distinct abscissas) from a vault for multiple sets $\mathrm{LOCK}(A^1, \ldots, A^l) = \{(x_i, S_i), i = 1, \ldots, N\}$, this leads to:

Lemma 2. *Assume that the sets $A^1, \ldots, A^l$ are randomly and uniformly chosen into $\mathcal{F}^t$. Given a vault $\mathrm{LOCK}(A^1, \ldots, A^l)$ with chaffing parameters ℓ, r, then for any $\mu > 0$, with probability at least $1-\mu$, there exist at least $\frac{\mu}{3} \ell^r q^{k-t} (\frac{r}{t})^t$ polynomials p of degree lower than k such that the Reed-Solomon codeword generated by p has t coordinates that are included in the vault.*

We remark an important difference compared with the use of l independent vaults (one for each A^j): instead of multiplying the number of possible polynomials by l, it is increased by a multiplicative factor ℓ^r (with $l \leq \ell$).

When using folded RS codes, the result does not hold. The analysis to estimate the number of polynomials is more complex as one should take in account the information given by the correlations within one column of evaluations. To have a generic security bound that is valid for folded Reed-Solomon codes, we estimate the entropy loss as for a secure sketch scheme:

Lemma 3. *The entropy loss of our Extended Fuzzy Vault scheme based on a $FRS_{\mathcal{F},\gamma,m,N,k}$ code is at most*

$$(mt-k)l \times \log_2 q - \log_2 \binom{r}{\lambda} + \log_2 \binom{N}{\lambda}$$

with $t \leq \lambda \leq lt$ is the number of indexing sets (the S_i's) covered by the genuine points.

Proof. Choosing the l evaluation functions corresponds to $l \times k \times \log_2 q$ bits of entropy (the choice of l polynomials). Let l_i be the size of each indexing set S_i after filling the vault only with genuine points (from the A^j's). Let λ the number of l_i strictly greater than 0. The choice of additional chaff positions requires $\log_2 \binom{N-\lambda}{r-\lambda}$ bits of randomness. The choice of the chaff values requires about $((r-\lambda) \times \ell + \sum_{i=1..N | l_i \neq 0} (\ell - l_i)) \log_2 q^m = m \times (r\ell - lt) \log_2 q$ (we assume here that ℓ is sufficiently small to neglect the difference between $\mathcal{F}^m$ and $\mathcal{F}^m \setminus \{F_d(x_i)\}_{d=1,\ldots,l}$).

Thus, we have approximately $\mathbf{H}_\infty(A^1,\ldots,A^l,R) = \mathbf{H}_\infty(A^1,\ldots,A^l)+\log_2 q\times (lk+m(r\ell-lt))+\log_2\binom{N-\lambda}{r-\lambda}$ where R denote the random bits that are used to lock the vault $\mathrm{LOCK}(A^1,\ldots,A^l)$.

Note that R is entirely determined by $A^1,\ldots,A^l$ and $\mathrm{LOCK}(A^1,\ldots,A^l)$, so $\overline{\mathbf{H}}_\infty(A^1,\ldots,A^l,R \mid \mathrm{LOCK}(A^1,\ldots,A^l)) = \overline{\mathbf{H}}_\infty(A^1,\ldots,A^l \mid \mathrm{LOCK}(A^1,\ldots,A^l))$. We know also that the output can be encoded with $\log_2\binom{N}{r}+mr\ell\log_2 q$.

This leads to $\overline{\mathbf{H}}_\infty(A^1,\ldots,A^l \mid \mathrm{LOCK}(A^1,\ldots,A^l)) \geq \mathbf{H}_\infty(A^1,\ldots,A^l)-(mt-k)l\times\log_2 q+\log_2\binom{N-\lambda}{r-\lambda}-\log_2\binom{N}{r}$.

We conclude from $\binom{N-\lambda}{r-\lambda}\binom{N}{\lambda}=\binom{N}{r}\binom{r}{\lambda}$. □

Note that we have at most $\mathbf{H}_\infty(A^1,\ldots,A^l)=l\log_2\binom{N}{t}$. Assuming that the sets are independent, this would give us a remaining entropy $\overline{\mathbf{H}}_\infty(A^1,\ldots,A^l \mid \mathrm{LOCK}(A^1,\ldots,A^l))$ greater than

$$\left(l\log_2\binom{N}{t}-\log_2\binom{N}{\lambda}\right)+\log_2\binom{r}{\lambda}-((mt-k)l\times\log_2 q)$$

This is somehow comparable with the remaining entropy when using l independent fuzzy vaults that can be approximated from Equation (1) as

$$l\left(\log_2\binom{r}{t}-(t-k)\log_2 q\right)$$

With t and λ small in front of N and r, the first equation can be approximated as $lN^t-N^\lambda+r^\lambda-(mt-k)l\times\log_2 q$ and the second equation is approximated as $lr^t-(t-k)l\times\log_2 q$. In both situation, r needs in general to be large to ensure that the entropy is large.

6 Implementation Issues

We discuss in this section some practical aspects of the Extended Fuzzy Vault scheme based on folded Reed-Solomon codes.

6.1 Noise

In [8], Juels and Sudan provide an example of their Fuzzy Vault scheme where the number of chaff points, to hide a given set, equals $q-t$. Transposed to our case, this would mean to have $N-t$ chaff points for each set with N up to q^m. But precisely as we deal with multiple sets encoded in the same vault, all of them would have the same N attributes x_i's, thus $\ell=N$, what is highly unrealistic since ℓ has to be smaller than m. We have therefore a restriction on the number of chaff points we can add for each set. Nevertheless, we could fill the space of possible symbols with noise up to ℓ for each of the N symbols (see Table 1).

x_0	x_1	x_2	x_3	x_4	x_5	...	x_{N-2}	x_{N-1}
$y_{0,1}$	$y_{1,1}$	$\times$	$y_{3,1}$	$y_{4,1}$	$y_{5,1}$		$y_{N-2,1}$	$y_{N-1,1}$
$y_{0,2}$	$y_{1,2}$	$\times$	$y_{3,2}$	$y_{4,2}$	$\times$	...	$\times$	$y_{N-1,2}$
$\times$	$y_{2,3}$	$\times$	$\times$	$\times$	$\times$		$\times$	$\times$

Table 1. An example of noise filling for $\ell = 3$

The lower amount of random chaff points per encoded set in the EFV scheme is compensated by the multiple sets themselves. The more there are sets, the more it is difficult for an attacker to find a specific set's attributes.

6.2 Complexity

List size Parameter s gives an upper bound on the size of the output list of algorithm from Section 3.3: recalling that the cardinality of $\mathcal{F}$ is q, output list is smaller than q^{s-1}. This was already stressed in [4] that this can be very large. In the EFV scheme, this is in particular dependent on the amount $l_i \leq l \leq \ell$ of users sharing a same information symbol x_i which can be quite large. As s has to be bigger than ℓ, we have to deal with potentially very big output lists.

Example 2 *Suppose the amount of referenced movies is $N = 10^4$, Alice has a list $(x_0, \ldots, x_{22})$ of her $t = 22$ favorites. Among the other users, we can easily imagine that two people love one common movie x_i with Alice ($l_i = 3$). Let us fix $\ell = 3$, $s = 4$ and $m = 5$ to tolerate up to $e = 3$ errors and $q > mN$. When Bob tries to learn who gets the same tastes as him, giving another set of 22 titles including the above movie x_i, the EFV scheme can be faced with a large output list.*

Note that in practice, our experiments have resulted into much smaller output lists.

Memory space and execution time Another point is that s and ℓ give a lower bound on m and determine the parameter D (8). The code size is directly impacted by these parameters, which leads to the memory space needed and the execution time. In particular, the size of the first linear system to solve grows fast and it is a critical element in the algorithm implementation.

Table 2 gives an overview of the size of the systems one has to solve while decoding. Implementation has been done using PARI/GP on a common desktop computer. Although the size is growing quickly, the execution time remains quite reasonable for these parameters.

t	m	s	ℓ	k	System size	Execution time
22	5	4	3	14	132*133	125 ms
50	10	8	6	15	900*905	14.5 sec
73	14	11	8	16	2336*2343	6 min 44 sec

Table 2. Some execution timings for EFV scheme with 100 users on $\mathcal{F} = \mathbb{F}_{2053}$

Vault size A vault obtained from locking multiple sets can be represented with $\log_2 \binom{N}{r} + mr\ell \log_2 q$ bits. As explained in Section 5, r needs in general to be large for security. This has an impact on the size of the vault representation.

It is important to remark that it is also the case for the original FV scheme (for which a vault can be represented with $\log_2 \binom{N}{r} + r \log_2 N$ bits).

Consequently, we underline another advantage of using our EFV scheme instead of l independent FV vaults (one for each A^j): If the number of collisions between the l sets $A^1, \ldots, A^l$ is small, then ℓ can be chosen small. The overall size of our EFV vault will be then about $\log_2 \binom{N}{r} + mr\ell(\log_2 N + \log_2 m)$ bits while the l vaults will need $l \times (\log_2 \binom{N}{r} + r \log_2 N)$ bits. For instance, if $N = r$, as soon $m\ell$ is much smaller than l, then the EFV vault size will become much smaller. In fact, this is due to the randomness that needs to be renewed for l independent vaults whereas we use the same for all sets in our EFV scheme.

7 Conclusion

In this paper, we extend the Fuzzy Vault scheme to handle multiple users at the same time. Our extension follows the quite recent improvements of the decoding properties of Reed Solomon codes as we are using list recovery methods for which effective algorithms have been discovered this year. Our first experiments prove that, today, direct applications of our proposal can be attempted. The next step would certainly to apply our Extended Fuzzy Vault scheme to biometric identification (in particular, for fingerprint modality). Due to the level of errors to handle, this does not seem directly feasible with current decoding techniques and would require further improvements.

References

1. Seung-Hoon Chae, Sung Jin Lim, Sang-Hyun Bae, Yongwha Chung, and Sung Bum Pan. Parallel processing of the fuzzy fingerprint vault based on geometric hashing. *TIIS*, 4(6):1294–1310, 2010.
2. Yongwha Chung, Daesung Moon, Sungju Lee, Seunghwan Jung, Taehae Kim, and Dosung Ahn. Automatic alignment of fingerprint features for fuzzy fingerprint vault. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *CISC*, volume 3822 of *Lecture Notes in Computer Science*, pages 358–369. Springer, 2005.

3. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer, 2004.
4. Venkatesan Guruswami. Linear-algebraic list decoding of folded Reed Solomon codes. In *IEEE Conference on Computational Complexity*, pages 77–85. IEEE Computer Society, 2011.
5. Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.
6. Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed Solomon and algebraic-geometric codes. In *FOCS*, pages 28–39. IEEE Computer Society, 1998.
7. Ari Juels and Madhu Sudan. A fuzzy vault scheme. In *Proceedings of IEEE Internation Symposium on Information Theory, ISIT*, Lecture Notes in Computer Science, page 408, 2002.
8. Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006.
9. Sungju Lee, Daesung Moon, Seunghwan Jung, and Yongwha Chung. Protecting secret keys with fuzzy fingerprint vault based on a 3d geometric hash table. In Bartlomiej Beliczynski, Andrzej Dzielinski, Marcin Iwanowski, and Bernardete Ribeiro, editors, *ICANNGA (2)*, volume 4432 of *Lecture Notes in Computer Science*, pages 432–439. Springer, 2007.
10. Jianjie Li, Xin Yang, Jie Tian, Peng Shi, and Peng Li. Topological structure-based alignment for fingerprint fuzzy vault. In *19th International Conference on Pattern Recognition (ICPR 2008), December 8-11, 2008, Tampa, Florida, USA*, pages 1–4.
11. Peng Li, Xin Yang, Kai Cao, Peng Shi, and Jie Tian. Security-enhanced fuzzy fingerprint vault based on minutiae's local ridge information. In Massimo Tistarelli and Mark S. Nixon, editors, *ICB*, volume 5558 of *Lecture Notes in Computer Science*, pages 930–939. Springer, 2009.
12. Peng Li, Xin Yang, Kai Cao, Xunqiang Tao, Ruifang Wang, and Jie Tian. An alignment-free fingerprint cryptosystem based on fuzzy vault scheme. *J. Network and Computer Applications*, 33(3):207–220, 2010.
13. Sung Jin Lim, Seung-Hoon Chae, and Sung Bum Pan. VLSI architecture of the fuzzy fingerprint vault system. In Friedhelm Schwenker and Neamat El Gayar, editors, *ANNPR*, volume 5998 of *Lecture Notes in Computer Science*, pages 252–258. Springer, 2010.
14. F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*. North-Holland Pub. Co., 1977.
15. Johannes Merkle, Matthias Niesing, Michael Schwaiger, Heinrich Ihmor, and Ulrike Korte. Performance of the fuzzy vault for multiple fingerprints. In Arslan Brömme and Christoph Busch, editors, *BIOSIG*, volume 164 of *LNI*, pages 57–72. GI, 2010.
16. Daesung Moon, Sungju Lee, Seunghwan Jung, Yongwha Chung, Miae Park, and Okyeon Yi. Fingerprint template protection using fuzzy vault. In Osvaldo Gervasi and Marina L. Gavrilova, editors, *ICCSA (3)*, volume 4707 of *Lecture Notes in Computer Science*, pages 1141–1151. Springer, 2007.
17. Abhishek Nagar, Karthik Nandakumar, and Anil K. Jain. Securing fingerprint template: Fuzzy vault with minutiae descriptors. In *19th International Conference on Pattern Recognition (ICPR 2008), December 8-11, 2008, Tampa, Florida, USA*, pages 1–4.

18. Karthik Nandakumar, Anil K. Jain, and Sharath Pankanti. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE Transactions on Information Forensics and Security*, 2(4):744–757, 2007.
19. Karthik Nandakumar, Abhishek Nagar, and Anil K. Jain. Hardening fingerprint fuzzy vault using password. In Seong-Whan Lee and Stan Z. Li, editors, *ICB*, volume 4642 of *Lecture Notes in Computer Science*, pages 927–937. Springer, 2007.
20. Atri Rudra. *List Decoding and Property Testing of Error Correcting Codes*. PhD thesis, University of Washington, 2007.
21. Adam Davison Smith. *Maintaining secrecy when information leakage is unavoidable*. PhD thesis, Cambridge, MA, USA, 2004. AAI0807529.
22. Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.
23. Umut Uludag, Sharath Pankanti, and Anil K. Jain. Fuzzy vault for fingerprints. In Takeo Kanade, Anil K. Jain, and Nalini K. Ratha, editors, *AVBPA*, volume 3546 of *Lecture Notes in Computer Science*, pages 310–319. Springer, 2005.