The ISC Int'l Journal of Information Security

January 2009, Volume 1, Number 1 (pp. 27–34)

http://www.isecure-journal.org

A Collusion Attack on the Fuzzy Vault Scheme

Hoi Ting Poon [a,*], Ali Miri [a]

[a] *School of Information Technology and Engineering, University of Ottawa, Canada, 800 King Edward Ave, Ottawa, Ontario, K1N 6N5, Canada*

ARTICLE INFO

Article history:
Received: 1 July 2008
Revised: 30 November 2008
Accepted: 20 January 2009
Published Online: 28 January 2009

Keywords:
Biometric Encryption, Fuzzy Vault, Vulnerability

ABSTRACT

The Fuzzy Vault scheme is an encryption scheme, which can tolerate errors in the keys. This leads to the possibility of enhancing the security in environments where these errors can be common, such as biometrics storage systems. Although several researchers have provided implementations, we find that the scheme is vulnerable to attacks when not properly used. This paper describes an attack on the Fuzzy Vault scheme where the attacker is assumed to have access to multiple vaults locked by the same key and where a non-maximal vault size is used. The attack effectively reduces the vault size by identifying and removing chaff points. As the vault size decreases, the rate at which chaff points are identified increases exponentially. Several possible defences against the attack are also discussed.

1 Introduction

In 1999, Juel and Wattenberg [1] introduced a cryptographic primitive called Fuzzy Commitment scheme to address the problem of secure biometric storage. This conceptually simple scheme makes use of error correction codes to allow for error tolerance in the key. In other words, the scheme allows a user to encrypt a message using a key K and decrypt the message with a key K' within a certain distance from K. This error tolerance capability allows for many interesting applications such as privacy-protected matching and the use of biometric features as keys in cryptosystems. However, the scheme has an important shortcoming. Geometric distortions such as rotations and translations are common in biometric readings. Hence, the key K' will likely consist of some permutation of the elements in K. The Fuzzy Commitment scheme does not tolerate such re-ordering of the symbols in the key [2].

To address this problem, Juel and Sudan [2] proposed the Fuzzy Vault scheme. By characterising the key as a set of symbols instead of a sequence and combining them with the proper error correction code, the scheme achieves the property of order-invariance. Recognizing its potential as a replacement for key release based biometrics systems, several researchers [3, 4] and [5] have already provided implementations of the Fuzzy Vault scheme. Studies have also emerged identifying potential flaws in chaff point placements [6, 7] and susceptibility to brute-force attack [8] in fingerprint applications. To improve the security of the scheme, some have proposed techniques such as key capsulation [9] and the addition of a password [10].

In this paper, we describe an attack on the Fuzzy Vault scheme where the attacker is given access to multiple vaults locked by the same key, and where the vault size is non-maximal. An example where such an attack is applicable would be the smartcard-based Fuzzy Vault system described by Clancy [3]. In this system, a typical user would carry multiple smartcards issued by different organizations, but the em-

* Corresponding author.
Email addresses: hpoon015@site.uottawa.ca (H.T. Poon), samiri@site.uottawa.ca (A. Miri).
ISSN: 2008-2045

bedded vaults would all be locked by the same key (the user's fingerprint). We also considered the case where multiple vaults are locked with the same secret message and found that a collusion attack is possible even when the vault is maximal. The situation can be difficult to avoid in many applications. Consider two of the applications proposed in [2]: privacy-protected matching and password recovery systems. An example of privacy-protected matching would be where a company has its contact information locked in a fuzzy vault with a set of product specifications such that only those who search for similar specifications would be able to unlock it. It is conceivable that a company would advertise its products through different companies/websites specialized in matching customers to sellers to increase exposure to the market. However, the specifications and contact information would remain largely the same. A typical password recovery system would be one where the user answers a number of questions such that a certain number of correct response would allow him to retrieve the password. Similarly for password recovery systems, users often maintain dozens of accounts. Many users often use the same password and many of the questions, and consequently answers, for password recovery are similar or identical, despite from different sites.

We will begin by presenting the Fuzzy Vault scheme. Then, we'll describe the collusion attack algorithms and explore some possible solutions to these attacks.

2 Fuzzy Vault Scheme

The Fuzzy Vault scheme contains two main algorithms: *Lock*, *Unlock*.

Setup For the functionality of Fuzzy Vault, one must first decide on a field $\mathbb{F}$ of order q and the range of values that the vault size, r, the message size, k, and locking set size, t, can take on.

Lock The *Lock* algorithm [2] is analogous to the encryption algorithm. Given a locking set (e.g. biometric features such as minutiae locations) $A = \{a_i\}_{i=1}^{t}$, where $a_i \in \mathbb{F}$, and a message, $M = \{m_i\}_{i=1}^{k}$, where $m_i \in \mathbb{F}$, to be locked, generates a set $V_A = \{(x_i, y_i)\}_{i=1}^{r}$, where $x_i, y_i \in \mathbb{F}$:

1: Create two empty sets, V_A and X
2: Set $Y(x) := m_1 + m_2 x + m_3 x^2 + \ldots m_k x^{k-1}$
3: **for** $i := 1$ to t **do**
4: $X := X \cup \{a_i\}$
5: $V_A := V_A \cup \{(a_i, Y(a_i))\}$
6: **end for**
7: **for** $i := t+1$ to r **do**
8: $x_i := U(\mathbb{F} \backslash \{X\})$, where U randomly chooses an element of the field $\mathbb{F}$ which is not part of X
9: $X := X \cup \{x_i\}$
10: $y_i := U(\mathbb{F} \backslash \{Y(x_i)\})$, where U randomly chooses an element of the field $\mathbb{F}$ which is not equal to $Y(x_i)$
11: $V_A := V_A \cup \{(x_i, y_i)\}$
12: **end for**

The resulting set V_A is called the Fuzzy Vault, locked under the set A.

As illustrated, the message is mapped to the coefficients of a polynomial, $Y(x)$, of degree $k-1$. The image of the locking set, $Y(A)$, is computed, producing a set of points $(a_i, Y(a_i))$ which lie on the polynomial $Y(x)$. To secure the message, we then add $r-t$ chaff points which are not part of the locking set and do not correspond to pairs that can be generated by $Y(x)$. Hence, any chaff point would not be a valid point on $Y(x)$ and cannot be confused with a point in the locking set. Note that q represents the size of the alphabet, r represents the size of the vault, t is the size of locking set and k is the minimum set overlap to achieve a match, where $q \geq r \geq t \geq k$. The vault size is maximal when it is equal to the size of the key space. The key space is defined here as the values that the x component can take on in the vault. In the description of the Fuzzy Vault scheme, the size of the key space is equal to the field size. However, in practice, it may be less than the field size. Also, the key space may be dependent on the locking set such as in [3], where a minimum distance is used to provide noise tolerance. Without loss of generality, the key space will be assumed to be equal to the field size for the remainder of the paper.

Unlock The *Unlock* algorithm [2] is analogous to the decryption algorithm. Given an unlocking set $B = \{b_i\}_{i=1}^{t_b}$, where $b_i \in \mathbb{F}$, and a vault, $V_A = \{(x_i, y_i)\}_{i=1}^{r}$, reproduce the message $M = \{m_i\}_{i=1}^{k}$ if $|A \cap B| \geq k$:

1: Generate an empty set Q
2: **for** $i := 1$ to t_b **do**
3: **if** $x_j = b_i$ **then**
4: $Q := Q \cup \{(x_j, y_i)\}$
5: **end if**
6: **end for**
7: $M := Decode(Q)$

The unlocking set B will contain some correct points and some chaff points. The Unlock algorithm identifies and retains the points in the vault where the x_i values match the unlocking set elements. If the sets A and B overlap significantly, the set Q will contain mostly the correct points that lie on the polynomial $Y(x)$. $Decode(Q)$ is a decoding operation that reconstructs the polynomial $Y(x)$, based on the t_b' overlapping points. More precisely, to reconstruct a polynomial of degree $k-1$, we need to have at least k points that lie on the polynomial. Hence, to reconstruct $Y(x)$, $|A \cap B| \geq k$. It is suggested that $Decode(Q)$ be implemented as a Reed Solomon

decoder since the set Q can be considered as a generalized Reed Solomon codeword [2, 11]. When using the classic Peterson-Berlekamp-Massey algorithm, the condition for successful decoding of the message becomes $|A \cap B| \geq (k + t'_b)/2$ [2].

A simple example of the algorithm is as follows: Let the message to be encrypted be $M = (1, 3, 2)$ and Alice's locking set be $A = (2, 5, 3, 1)$. Then, $Y(x) = 1 + 3x + 2x^2$. We first compute $Y(A)$, producing the set of points $(2, 15), (5, 66), (3, 28), (1, 6)$. Two chaff points $(4, 5), (6, 50)$ are then added to construct the vault, $V_A = \{(1, 6), (2, 15), (3, 28), (4, 5), (5, 66), (6, 50)\}$ where $Y(4) \neq 5$ and $Y(6) \neq 50$. If Bob provides the unlocking set $B = (4, 2, 3, 5)$ to unlock the vault, the algorithm would attempt to reconstruct $Y(x)$, given $Q = \{(2, 15), (3, 28), (4, 5), (5, 66)\}$, and succeed because A and B overlap on more than $k = 3$ points. However, if Bob provides an unlocking set $B = (2, 4, 3, 6, 7)$, he would fail to unlock the message since the set overlaps on only 2 points [4].

2.1 Security

The security of the Fuzzy Vault scheme relies on the chaff points' ability to hide the true points. The effect of the chaff points is to increase the number of spurious polynomials. A spurious polynomial is a curve of degree k or less containing a set of exactly t points in the vault, which is not the secret polynomial. In the information theoretic sense, the security can be defined as the number of plaintexts associated with a ciphertext. For the Fuzzy Vault scheme, this translates to the number of valid messages for a particular vault. Consider a vault V_A locked under the set A, where there are r sets of points, of which t are real. Then, there exists at least $\frac{\mu}{3} q^{k-t} (\frac{r}{t})^t$ spurious polynomials, with probability at least $1 - \mu$, for $\mu > 0$ [2]. For example, for $\mu = 0.01, r = 300, t = 22, k = 19, q = 2^{16}$, we would have at least $108844773 \approx 2^{26.7}$ spurious polynomials, with probability 0.99 [2]. Assuming any point is equally likely to be a true point, the attacker would have to consider all these polynomials to equally likely be the secret polynomial. The system can be said to achieve 26-bit security at the 99% confidence level.

One can easily show that this algorithm is vulnerable to known plaintext attacks. If an attacker can gain access to the message of a vault, the locking set, i.e. the key, can be easily obtained by verifying the message polynomial on the points in the vault. The following algorithm illustrates such an attack:

Given a vault V_A locked under the set $A = \{a_i\}_{i=1}^{t}$ with the message $M = \{m_i\}_{i=1}^{k}$, where each vault contains r points, $V_A = \{(x_i, y_i)\}_{i=1}^{r}$ and where $X = \{x_i\}_{i=1}^{r}$, perform the following:

1: Create an empty set X'
2: Set $Y(x) := m_1 + m_2 x + m_3 x^2 + \ldots m_k x^{k-1}$
3: **for** $i := 1$ to r **do**
4: **if** $y_i = Y(x_i)$ **then**
5: $X' := X' \cup \{x_i\}$
6: **end if**
7: **end for**
8: X' is the locking set A

The sets X' and A are equal because all true points (x_i, y_i) lie on the polynomial $Y(x)$ while the chaff points do not.

Any practical cryptosystem must at least be resistant to ciphertext-only attacks. However, we will show below that the Fuzzy Vault scheme is vulnerable to these attacks if it is not properly used.

3 Collusion Attack

We first consider the case where the attacker has access to multiple vaults locked by the same key and where the vault size is non-maximal (i.e $r < q$). We found that the attacker can reduce the number of candidate polynomials by exploiting the following properties:

- The keys used to lock the vaults are the same and 'visible' to the attacker
- The chaff points are generated randomly and are independent of the key
- Since the vault size is non-maximal, the chaff points vary from vault to vault

In the collusion attack algorithm, the goal of the attacker is to identify and remove chaff points, thus, reducing the number of spurious polynomials. The key can even be revealed when a sufficiently large number of vaults are available.

3.1 Collusion Attack Algorithm with $A_i = A$

Given n vaults $\{V_{A,1}, V_{A,2} \ldots V_{A,n}\}$ locked under A, where each vault contains r points, $V_{A,j} = \{(x_{i,j}, y_{i,j})\}_{i=1}^{r}$ and where $X_j = \{x_{i,j}\}_{i=1}^{r}$, perform the following:

1: Create an empty set X'
2: **for** $i := 1$ to r **do**
3: **if** $x_{i,1} \in X_j$ for all j **then**
4: $X' := X' \cup \{x_{i,1}\}$
5: **end if**
6: **end for**
7: $V_{A,j}^{eff} := \left\{(x_{i,j}, y_{i,j}) \in V_{A,j} | X_{i,j} \in X'\right\}$, denoted the j^{th} effective vault
8: $r_{eff} := \left|V_{A,j}^{eff}\right| = |X'|$, denoted the effective vault size

In other words, the algorithm searches for x values which appear in all the vaults and retain them as possible elements in the true sets. Points that do not ap-

pear in all vaults are identified as chaff points, and are removed. The effective vault is the result of this sieving process. As the number of vaults increases, more chaff points are identified and the set X' approaches the set A since the set A must appear in all n vaults while the randomly generated chaff points may not. Note that the attack is applicable as long as there are some common attributes used to produce the locking sets. It is not required for all elements in the locking sets to be in common and the use of different fields among the n available vaults do not protect against the attack as long as the mapping from the attributes to the locking sets are known.

3.2 Collusion Attack Algorithm with $A_i = A$ and $M_i = M$

If the attacker can obtain two or more vaults locked with the same key and with the same secret message, the key can be revealed even more easily using similar ideas as the previous algorithm.

Given n vaults $\{V_{A,1}, V_{A,2} \dots V_{A,n}\}$ locked under the same set $A = \{a_i\}_{i=1}^{t}$ with the same message $M = \{m_i\}_{i=1}^{k}$, where each vault contains r points, $V_{A,j} = \{(x_{i,j}, y_{i,j})\}_{i=1}^{r}$, perform the following:

1: Create an empty set V_{eff}
2: **for** $i := 1$ to r **do**
3: **if** $(x_{i,1}, y_{i,1}) \in V_{A,j}$ for all j **then**
4: $V_{eff} := V_{eff} \cup \{(x_{i,1}, y_{i,1})\}$
5: **end if**
6: **end for**
7: $M' := Decode\left(V_{eff}\right)$
8: V_{eff} is the effective vault
9: $r_{eff} := \left|V_{eff}\right|$ is the effective vault size

Since $x_{i,j}$ and $y_{i,j}$ are related by a common polynomial $Y(x) = m_1 + m_2 x + m_3 x^2 + \dots m_k x^{k-1}$, these points must be identical across all vaults. If the chaff points are generated randomly, the probability of two chaff points from different vaults being identical is very low, assuming a reasonably large field size. Hence, the attacker will likely be able to identify A even in the case where $n = 2$. Note that, unlike the previous case, the attacker would succeed even when the vault size is maximal. When the maximum vault size is used, the chaff points in all the vaults contain the same x_i values, but the y_i values they are matched with are randomized while the true points have fixed x_i and y_i. Hence, an attacker can exploit this property to identify chaff points.

Both algorithms assume that all n vaults are locked by the very same key. However, when using Fuzzy Vault for biometric applications, the keys used to lock the vaults are not likely to be identical; rather they overlap significantly. In this case, the algorithms can be modified such that $x_{i,j}$ or $(x_{i,j}, y_{i,j})$ appearing in the majority of the vaults are retained. In Clancy's smart-card implementation [3], a minimum distance is also used to avoid confusion between neighbouring $x_{i,j}$. Then, we simply adapt by considering $x_{i,j}$ and $x_{i,j'}$ to be the same if they are within the minimum distance from each other.

3.3 Collusion Attack Algorithm with $M_i = M$

Finally, we consider the case where the attacker is given n vaults locking the same message, but with different keys. If the probability distribution of the locking sets is well defined, then the probability of a particular point being a true point is fixed. We define the probability of a point x being in the locking set as p_t. Then, with probability p_t, the point $(x, Y(x))$ would appear in the vault. When x is not in the locking set, the probability of it being chosen as a chaff point is $p_{chaff} = c/s$, where c is the number of chaff points to be added and s is the number of possible values for a chaff point. Then, with probability $(1 - p_t)p_{chaff}$, we would have (x, y') in the vault, where y' is uniformly distributed over $\mathbb{F} \backslash \{Y(x)\}$. For clarity, we will assume the vault is maximal, i.e. $p_{chaff} = 1$. Hence, $P(x, y' = y) = (1 - p_t)/(q - 1)$, where y is a value other than $Y(x)$. For $p_t \gg 1/q$, we can see that $p_t \gg (1 - p_t)/(q - 1)$. The analysis follows similarly when $p_{chaff} \neq 1$.

Assuming there are $N \gg k$ points where $p_t \gg 1/q$, a collusion attack would proceed as follows:

Given n vaults $\{V_1, V_2 \dots V_n\}$ with the same message $M = \{m_i\}_{i=1}^{k}$, where each vault contains r points, $V_j = \{(x_{i,j}, y_{i,j})\}_{i=1}^{r}$, perform the following:

1: Create a q by q table $Count(x, y)$ and initialize all values to 0.
2: **for** $j := 1$ to n **do**
3: **for** $i := 1$ to r **do**
4: $Count\left(x_{i,j}, y_{i,j}\right) := Count\left(x_{i,j}, y_{i,j}\right) + 1$
5: **end for**
6: **end for**
7: $Count(x, y)$ contains the distribution of data over the n vaults
8: Create an empty set V_t
9: **for all** $a \in \mathbb{F}$ **do**
10: Find the mean u_a of $Count(a, y)$
11: The estimated true point is (a, b) where $b := y'$ where $|Count\left(a, y'\right) - u_a|$ is maximal
12: $V_t := V_t \cup \{(a, b)\}$
13: $Q(a) := |Count(a, b) - u_a|$
14: **end for**
15: V_t is the estimated true vault, which contains the points most likely to lie on the message polynomial.
16: $Q\left(a\right)$ is a measure of the quality of the point retained.
17: $V_{eff} := m$ pairs of (a, b) with the highest $Q\left(a\right)$
18: $M' := Decode\left(V_{eff}\right)$

If the probability of a point being true is high, it will be consistently paired with a particular y value.

The algorithm exploits this property to identify a set of candidate true points. When there are a large number of vaults available, the points where $p_t \ll 1/q$ may also be useful as they will appear not to be ever paired with a particular y value. To identify which estimated true points are most likely to be correct, we simply note that the frequency of the ones which appear much greater or much less than its peers. For example, if we see $(1,5),(1,5),(1,7),(1,5),(1,5)$ across five vaults, it would be highly probable that $(1,5)$ is a true point. If we see $(5,3),(5,4),(5,8),(5,8),(5,2)$ across five vaults, the point $(5,8)$ only appears slightly more often than its peers and, thus, is a less reliable candidate for a true point.

In both cases where $M_i = M$, the situation becomes more complex when the field differs from vault to vault. Although a direct comparison may not be possible, the question of whether it is possible to narrow down the candidate true points remains open.

3.4 Security

In the algorithms from sections 3.1 and 3.2, the goal is to identify and remove chaff points to produce a smaller vault, V_{eff}, which still contains all the true points. Hence, we can think of the security of the system, when n vaults are used to be equivalent to that of a single vault V_{eff}. Recall from the security analysis in section 2.1 that there are at least $\frac{\mu}{3}q^{k-t}\left(\frac{r}{t}\right)^t$ spurious polynomials in a vault, with probability $1-\mu$. When generating V_{eff}, the only parameter that was changed in favour of the attacker is the vault size, r_{eff}. The incurred security loss is, thus, $\left(\frac{r}{r_{eff}}\right)^t$.

To better understand the degree at which the security deteriorates, we need to examine the rate at which r_{eff} decreases with respect to the number of vaults available, n, and the vault size, r.

Availability of two vaults locked with the same key is the most likely situation to occur in practice, where the key may, for example, be a person's fingerprint. If we have two vaults locked with the same key and different messages with t true points and $r-t$ random chaff points, all the true points must appear in both vaults while the probability that x chaff points match between the vaults is given by:

$$P(x) = \frac{\binom{c_1}{x}\binom{s-c_1}{c_2-x}}{\binom{s}{c_2}} \quad \text{where} \quad s = q-t, \quad c_1 = r_1 - t, \quad c_2 = r_2 - t \tag{1}$$

Here, s represents the chaff space, and c_1 and c_2 are the number of chaff points in the first and second vault. If we are given $i+1$ vaults with the same parameters, the probability that x_{i+1} chaff points match in all these vaults is given by the following iterative function:

$$P(x_{i+1}) = \sum_{x_i=1}^{c} P(x_i)\frac{\binom{c}{x_{i+1}}\binom{s-c}{x_i-x_{i+1}}}{\binom{s}{x_i}} \quad \text{where} \quad P(x_1) = \begin{cases} 1 & x_1 = c \\ 0 & else \end{cases} \tag{2}$$

The expected effective vault size of $i+1$ with the parameter set (r,t,q) is thus the sum of the number of true points and the expected number of matches across the vaults:

$$E\{r_{eff}\} = t + \sum_{x_{i+1}=1}^{c} x_{i+1}P(x_{i+1}) \tag{3}$$

The variance of r_{eff} can be used as a measure of the accuracy of the mean value.

When $s-c$ is small and s is large, the rate at which $E\{r_{eff}\}$ decreases with respect to n is almost linear, approximately $(s-c)$/vault. This is intuitive since two sets containing almost the entirety of the universe will surely overlap significantly. The missing elements from either set will contain a small portion of the universe and, when chosen randomly, will almost surely be different. However, when $s-c$ is large, $E\{r_{eff}\}$ decreases exponentially with respect to n. From the equation, we can see that the term $\binom{s-c}{x_i-x_{i+1}} = 0$ for all $x_i - x_{i+1} > s-c$. Hence, the effective vault size can at most decrease by $s-c$. In short, the parameters s and c allow us to control the rate at which the effective vault size decreases.

Table 1 shows the deterioration of security in terms of the number of spurious polynomials, N_{sp}, given n vaults with the same locking set for the following parameters: $q = 2^8, t = 20, k = 16, \mu = 0.01$. The two cases shown, $c = 200$ and $c = 180$, highlight the effect of the number of chaff points on the rate at which security decreases. It can be verified that $P(|r_{eff} - E\{r_{eff}\}| < 8) > 90\%$ for $n \leq 10$ in both cases.

If two vaults contain the same message and are locked with the same key, the situation becomes more alarming. Since the true points are guaranteed to appear in both vaults, we cannot mistake a true point as a chaff point. The only source of error would be in identifying a chaff point as a true point, which would occur when the randomly generated x and y values match. The probability of a match in the y values for two chaff points is $1/(q-1)$. Thus, using the previous equation for $P(x)$ to evaluate the probability of matching x values, we find that the probability of k chaff points matching between two vaults is

Table 1. Expected effective vault size and corresponding security given n vaults with the same locking set

c = 200				c = 180			
n	**E** $\{\mathbf{r}_{eff}\}$	**N** $_{sp}$	**Bits of security**	**n**	**E** $\{\mathbf{r}_{eff}\}$	**N** $_{sp}$	**Bits of security**
1	220	$5.22 * 10^8$	28.96	1	200	$7.76 * 10^7$	26.21
2	189	$2.50 * 10^7$	24.58	2	157	$6.13 * 10^5$	19.23
3	164	$1.47 * 10^6$	20.48	3	125	6420	12.65
4	142	82241	16.33	4	100	74	6.2
5	123	4650	12.18	5	81	1	0
6	107	286	8.16	6	66	0	-
7	94	21	4.42	7	55	0	-
8	83	2	1	8	47	0	-
9	73	0	-	9	40	0	-

$$P(k) = \sum_{x=k}^{c} P(x) \left(\frac{1}{q-1}\right)^k \left(1 - \frac{1}{q-1}\right)^{x-k} = \left(\frac{1}{q-1}\right)^k \sum_{x=k}^{c} P(x) \left(1 - \frac{1}{q-1}\right)^{x-k}. \quad (4)$$

For q large and $k > 0$, $(\frac{1}{q-1})^k$ is very small and $(1 - \frac{1}{q-1})^{x-k} \approx 1$. Thus, $P(k) \approx (\frac{1}{q-1})^k \sum P(x)$. Hence, with very high probability, we would have no matches $(k = 0)$ between the two vaults.

The algorithm described in section 3.3 exploits the possibility that the locking set is distributed such that a true point would appear more frequently than a chaff point. Note that the condition allowing the identification of the true points, $p_t \gg 1/q$, is not difficult to satisfy since a large field size is often used and the required k points is much less than the points available in a vault. The distribution of biometric features would certainly satisfy this condition. In fingerprints, minutiae locations are most prominent near the core and rarely at the edges. The question of how many vaults would be needed to compromise the scheme depends on various factors, particularly the distribution of the locking sets and the degree of the message polynomial. However, if q is large, it is unlikely that the two chaff points will have matching y value. The probability of identifying a true point given n vaults can then be approximated by the probability that x is a true point in at least two vaults: $1 - (1 - p_t)^n - np_t(1 - p_t)^{n-1}$. The number of vaults needed to identify a true (x, y) pair is then approximately $2/p_t$.

3.5 Example of Collusion Attack

Uludag [4] implemented the Fuzzy Vault scheme using fingerprints. To produce the locking set, the minutiae coordinates of each fingerprint image are first quantized such that they lie in a square tessellation of 7 pixels width. The coordinates are then expressed in 8-bit values such that the range of x or y would be $[0, 255]$. The locking set is then defined to be the set $A = \{a_i\}_{i=1}^t$ where $a_i = x_i|y_i$, over the field $GF(2^{16})$. The number of true points, t, is selected to be 18 and the number of added chaff points is 200.

Assuming that an attacker has access to two vaults from the same user, we can estimate the effective vault size using the previously found equations, where $s = 2^{16} - 18$, $c = 200$. With almost 90% probability, the two vaults would have at most one chaff point in common. Thus, the key and the message will probably be revealed immediately.

Actually, the number of chaff points per vault is also optimistic. The size of the fingerprint images from the database is approximately $300 * 400$. With a block size of 7-pixel width, this translates to at most $43 * 58$ possible minutiae coordinates. Hence, out of the $2^{16} = 65536$ possible values for $x|y$, only $43 * 58 = 2494(\approx 4\%)$ are valid. Since the chaff points were drawn randomly from $GF(q)$, only 8 chaff points on average would be valid minutiae coordinates. Then, a simple brute force attack would be able to identify the key and the message. Hence, one must be careful and generate chaff points from the key space and not from the field when the size of the key space is less than that of the field.

4 Modifications to the Scheme

There are several possibilities in modifying the scheme to defend against a collusion attack. We concentrate on the case when the same locking set is used to produce multiple vaults and briefly investigate three possible modifications pertaining to each of the exploitable properties discussed at the beginning of section 3: a) 'visibility' of key elements, b) Variability of chaff points across vaults and c) non-maximal vault size.

4.1 One-Way Transform of the Locking Set

The collusion attack is possible partly because the true points don't change and they are immediately accessible as x_i values. Thus, one way to protect against the attack is to generate different sets of x_i values for different vaults even when the locking sets are the same. Obviously, we must also not allow the attacker to retrieve the locking set elements from x_i as he would then simply reverse the operation and carry out the attack.

The idea is similar to that of cancellable biometrics [12] or password salting. We apply a keyed one-way transform on the locking set for each vault. If the attacker cannot reverse the one-way transform, he cannot retrieve x_i. In other words, we compute $a'_i = F(a_i, S)$ for all $a_i \in A$, where $F(x)$ is a one-way function such as a cryptographic hash function, and S is the salt consisting of a random bit string. Then, we proceed with the Fuzzy Vault scheme with $a'_i \in A'$. Since a different S is used for different vaults, the true sets will vary and, thus, the collusion attack would fail.

When the key space is large, one can simply store S along with the vault since it does not help the attacker in retrieving the set A. By using this technique, the security achieved is thus that of the difficulty of computing the preimage set of $F(a_i, S)$, and of identifying the correct polynomial in a vault of size r_{eff}, provided that it's less than the security of a single vault.

When the key space is small, a brute force attack would easily provide a complete mapping for all possible a_i. Unfortunately, the entropy of most biometric features tends to be small. A one-way transform of these feature points would provide minimal security enhancement. One possible solution is to use S as part of the key (e.g. PIN number). In other words, we increase the key space via the use of a consistent password. The security gained is simply the number of bits in S.

4.2 Deterministic Chaff Point Generation

Rather than varying the true points in different vaults, we can also fix the chaff points so that different vaults locked by the same key would have the same set of x_i values. In other words, we generate the chaff points dependent on the locking set A. To do so, we need to produce a consistent bit string from the locking set A. The bit string is then used to initialize a pseudo-random generator, resulting in a consistent set of chaff points. We note that the problem of producing a consistent bit string from A is similar to that of retrieving the message from a potentially corrupt codeword. Hence, we propose the use of error correction codes. The locking set A can be considered a codeword with some symbol errors. By choosing the appropriate parameters, we can perform $Decode'(A)$ to generate a consistent A'. We then initialize a pseudo-random generator to A' and generate the chaff points accordingly.

Note that the size of A' will necessarily be smaller than A. If the size of A is not large enough or if it requires a significant amount of error tolerance, the size of A' may become small enough to allow an attacker to perform a brute force attack. By running through all the possibilities for A', he would be able to match a set of chaff points to a set $A = Encode'(A')$. Since a vault consists almost entirely of chaff points, he would then be able to deduce the locking set A.

4.3 Maximal Vault Size

By using the maximal vault size, the vault will contain all the possible x_i values regardless of the locking set. Hence, the chaff points do not vary. As long as the same message is not used twice with the same locking set, we can ensure that a collusion attack would not succeed.

5 Conclusion

We demonstrated an attack on the Fuzzy Vault scheme when an attacker can gain access to multiple vaults locked with the same key, where the vault size is non-maximal. Our analysis showed that, as the vault size decreases, a collusion attack could quickly reduce the security of the scheme. Some possible defences against the attack were suggested, but they are highly dependent on the property of the locking set. Due to the low entropy of biometrics, it may be ideal to always use the maximal vault size.

References

[1] Ari Juels and Martin Wattenberg. A Fuzzy Commitment Scheme. In *Proceedings of the 6th ACM Conference on*

Computer and Communications Security (CCS'99), pages 28–36, Kent Ridge Digital Labs, Singapore, 1999. ACM Press.

[2] Ari Juels and Madhu Sudan. A Fuzzy Vault Scheme. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, page 408, 2002.

[3] T. Charles Clancy, Negar Kiyavash, and Dennis J. Lin. Secure Smartcard-Based Fingerprint Authentication. In *Proceedings of the ACM SIGMM Workshop on Biometrics Methods and Applications (WBMA'03)*, pages 45–52, Berkley, California, 2003. ACM.

[4] Umut Uludag, Sharath Pankanti, and Anil K. Jain. Fuzzy Vault for Fingerprints. In *Proceedings of the Audio- and Video-based Biometric Person Authentication (AVBPA'05)*, pages 310–319, Hilton Rye Town, NY, USA, 2005.

[5] S. Yang and I. Verbauwhede. Automatic Secure Fingerprint Verification System Based on Fuzzy Vault Scheme. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, pages 609–612, Philadelphia, PA, USA, 2005.

[6] Ee-Chien Chang, Ren Shen, and Francis Weijian Teo. Finding the Original Point Set Hidden among Chaff. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*, pages 182–188, Taipei, Taiwan, 2006. ACM.

[7] Ee-Chien Chang and Qiming Li. Hiding Secret Points Amidst Chaff. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (Advances in Cryptology - EUROCRYPT'06)*, volume 4004 of *Lecture Notes in Computer Science (LNCS)*, pages 59–72, Petersburg, Russia, 2006. Springer.

[8] Preda Mihailescu. The Fuzzy Vault for Fingerprints is Vulnerable to Brute Force Attack. http://arxiv.org/abs/0708.2974v1.

[9] W.L.W. AlTarawneh and W.L Woo. Biometric Key Capsulation Technique Based on Fingerprint Vault: Anatomy and Attack. In *Proceedings of the International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA'08)*, pages 1–5, Damascus, Syria, 2008.

[10] Karthik Nandakumar, Abhishek Nagar, and Anil K. Jain. Hardening Fingerprint Fuzzy Vault Using Password. In *Proceedings of the International Conference on Biometrics (ICB'07)*, volume 4642 of *Lecture Notes in Computer Science (LNCS)*, pages 927–937, Seoul, Korea, 2007. Springer.

[11] S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *International Journal of Applied Mathematics*, 8(2):300–304, 1960.

[12] N. Ratha, J. Connell, and R. Bolle. Enhancing Security and Privacy in Biometrics-Based Authentication Systems. *IBM System Journal*, 40(3):614–634, 2001.

Hoi Ting Poon received his BSc and MSc in Electrical Engineering from the University of Ottawa in 2005 and 2007 respectively, where he is currently a PhD candidate. His research interests include information security, system security, cryptography, biometric encryption and authentication systems.

Ali Miri received his BSc and MSc in Mathematics from the University of Toronto, Canada, and his PhD in Electrical and Computer Engineering from the University of Waterloo, Canada. Since 2001, he has been with the School of Information Technology and Engineering (SITE) at the University of Ottawa, Canada, where he is also the director of the Computational Laboratory in Coding and Cryptography (CLiCC). His research interests include digital communication, and applied security. He is a member of Professional Engineers Ontario, ACM and a senior member of IEEE.