Fuzzy Vault for Behavioral Authentication System

Md Morshedul Islam(✉) and Reihaneh Safavi-Naini

University of Calgary, Calgary, Canada
{mdmorshedul.islam,rei}@ucalgary.ca

Abstract. A fuzzy vault encrypts a message using fuzzy data such as user's biometric data as the vault key. Fuzzy vault can be used to protect users' cryptographic keys in smart cards and inside applications. We consider fuzzy vault based on behavioral data. A behavioral profile of a user consists of a set of features that collectively authenticates the user. Compared to biometric vault behavioral vault has the advantages of being revocable and less privacy sensitive. Fuzzy vaults for behavioral data, however, introduces significant challenges including feature representation, and feature matching algorithms that can provide the required correctness, security, and efficiency. We design and analyze a fuzzy vault based on the user's behavioral data that employs a novel soft-decision decoding algorithm and implement our design for two behavioral authentication (BA) systems. Our approach is general and can be used for other BA systems. We discuss our results and directions for future research.

Keywords: Fuzzy vault · BA system · BAVault

1 Introduction

Fuzzy vault was proposed by Juels and Sudan [12] to encrypt (*lock*) a sensitive value into a cryptographic *vault* such that unlocking needs a decryption key that is "close" to the encryption key. In a fuzzy vault, the sensitive data (e.g., a cryptographic key) defines a polynomial $f(x)$ of degree k over a finite field $\mathbb{F}_q$, that will be evaluated for each element of a "locking" set $A \subset \mathbb{F}_q$, to form a set of "legitimate points", $L = \{a_i, f(a_i) : a_i \in A\}$. The set L is then combined with a set Ch of *chaff points* that are randomly selected from $\mathbb{F}_q^2$ and used to hide the elements of L in the vault (ciphertext) $V = L \cup Ch$. The vault can be "opened" (and the sensitive message recovered) by using an unlocking set B, where B has a large overlap with A. Biometric-based fuzzy vault uses a user's biometric features for the locking and unlocking sets. Fuzzy vaults have been implemented using fingerprint, iris, and face data [6,14,15,17,21], and for higher security, using multimodal biometric systems [13]. Fuzzy vaults have been used for biometric-based protection of cryptographic keys in smart cards [6] and online authentication systems [24]. In these applications, the secret key is stored in the fuzzy vault and becomes available to the system when the user presents their correct biometric. Fuzzy vault provides an attractive solution for the protection of keys in mobile apps and electronic wallets.

Juels et al. proved that recovering the correct polynomial without knowing the set A, has exponential computation in the number of chaff points. Biometric implementations of fuzzy vault, however, introduce new attacks due to imperfect selection of chaff points, and for the attacks on the underlying authentication systems. Protection against these attacks together with providing efficiency has been widely studied [6,21,22]. Using biometric-based fuzzy vaults in multiple applications introduces a new threat: if multiple vaults of the same user are leaked, the user biometric data will be compromised [19]. This will allow all the previous vaults to be opened, and also the future usage of the fuzzy vault for that user becomes insecure. So, it is important to make the bio-data of the user changeable (*revocable*). We achieved this goal by using behavioral data and referred to such vaults as BAVault. A Behavioral Authentication (BA) system [3, 8,10] constructs a profile for the users by identifying and capturing a number of behavioral features during well-designed activities. A behavioral profile consists of a set of d features, each represented by a set of n samples, forming a $n \times d$ matrix. A BAVault will use a user's profile to lock a secret, and its *unlocking algorithm* will "match" the profile of a verification claim to unlock the vault.

Advantages of BAVaults compared to biomteric based systems are less linkability of behavioral data, revocability by replacing the underlying BA system with a similar system, and no requirement for additional hardware.

Challenges of Implementing a BAVault. The main challenge of BAVaults is the high dimensionality of feature data that results in inefficient implementations. In a BA system, a feature consists of n sample vectors and direct mapping of a behavioral feature to a finite field results in a high degree extension field (e.g. in DAC[10] a sample feature vector has dimension 40–120). Generating (chaff points), locking and unlocking over a field of this size result in an extremely inefficient system (see Section 4).

Our Work. In BA systems, a feature corresponds to a probability distribution. We represent a feature with its first and the second moments (mean and variance). We use the samples in the profile and the verification claim to construct estimates of these moments. The task of feature matching is to decide if the two pairs of estimates correspond to the same underlying distribution. This allows us to represent a feature as a tuple of two elements of $\mathbb{F}_q$, and will result in a vault that is as efficient as a fingerprint vault (i.e. using a field extension of degree 2 or 3) [6,17,21]. It, however, requires a well-designed decoding approach to reduce the error in matching. A major challenge in using this compact representation is generating chaff points. In a BAVault, a feature is mapped to a point $(a_i, f(a_i))$, where a_i is obtained from the mean and variance of the feature. A chaff point is of the form $(a_i, \bar{a_i})$ where a_i represents the mean and variance of a hypothetical distribution, and $\bar{a_i} \neq f(a_i)$. The value of a_i in chaff point must be chosen such that its corresponding distribution will be distinguishable for the distributions of all features using the chosen statistical tests. Additionally, the set of chaff points should not "separable" from the set of vault points $(a_i, f(a_i)$ which correspond to true features.

Figures 1a and 1b are the block diagrams of the proposed locking and unlocking algorithms of the BAVault, respectively. The algorithms follow the structure of biometric vault (e.g., Fig. 2 in [21]), with the new components indicated with dashed lines. In a BAVault, the secret message is first encoded using Cyclic Redundancy Check (CRC) and used to define a polynomial $f(x)$. The features are preprocessed using a *random projection (RP)* algorithm that transforms the feature set into a smaller set of features, each a random linear combination of the original features. Each projected feature is then represented by a pair $a_i = (\mu_i, \sigma_i^2)$ of the mean and variance of the transformed feature which will be used as the evaluation points for $f(x)$. A *chaff point* $(a_i, \bar{a_i})$ is generated through a multi-step process such that a_i corresponds to a hypothetical distribution with mean and variance (μ, σ^2), which is not "close" to real feature where closeness is measured by using statistical tests for mean (`tTest` [16]) and variance (`fTest` [9]). The *unlocking algorithm* is a novel *soft-decision decoding* algorithm (details is in Sect. 4).

We evaluate the security of the vault in protecting, (i) the secret message, and (ii) the privacy of the profile. We consider the following attacks that have been used to evaluate the security of biometric vault systems: (i) *chaff point* recovery attack that uses uneven distribution of vault points in the vault space allowing the attacker to infer the correct feature points, (ii) using multiple vaults with the same user profile that allows the attacker to recover the true features by comparing the set of vault points, and (iii) impersonation through mimicry attack against the underlying BA system.

BAVault Implementation and Experiments. We implemented and evaluated our design on two BA systems. The first system is *Touchalytics* [8] that uses touch behavior such as up-down and left-right scrolling to verify the user's identity. The second system, *DAC (Draw A Circle)* [10], is a challenge and response system that verifies the users' identity using their behavior in drawing challenge circles. We designed a feature-based verification algorithm and tuned the parameters to ensure a lower error rate. The collected data was also used to implement and evaluate the BAVault. We used the published data for Touchalytics, and collected new data for the redesigned DAC. Through extensive experiments, we show that (i) both BAVaults have acceptable error rate; (ii) a random invalid claim can recover a very small number of legitimate points, while a mimicry attack can recover more feature points. However, in both cases, the recovered features are not sufficient to unlock the vault. We also show that RP prevents the multi-vault attack, and ensures that in a compromised vault user's profile will not be leaked. We estimate the number of polynomials that a brute force attacker must check to achieve success. The secret message sizes in Touchalytics and DAC based *BAVault* are 192–208 bits and 448–480 bits, respectively.

Ethics Approval. We obtained ethics approval from the Research Ethics Board of our institution and performed our experiments in accordance with ethics guidelines governing user personal data privacy and security.

Paper Organization. Section 2 is background and related works. Section 3 defines BAVault and Sect. 4 gives the design of our BAVault. Section 5 is implementation and experimental results and Sect. 6 concludes the paper.

2 Preliminaries and Related Works

We recall definitions and results that are used in the remainder of the paper.

BA System. A BA system constructs a *behavioral profile* for a user, which consists of a set of features measurements, and uses it to verify verification request, a second set of measurements of the features. A BA profile $\mathbf{X} = (F_1, F_2, \cdots, F_d)$ consists of d features, and can also be represented by a set $\{\mathbf{x_i}, i = 1, \cdots, n\}$ of n vectors of dimension d over $\mathbb{R}^d$. The set $\{x_{i,j}, i = 1, \cdots, n\}$ grips n samples of a feature F_j, represents intrinsic behavioral characteristics of a user.

Matching Algorithm. A matching algorithm $M(\mathbf{X}, \mathbf{Y})$ compares *verification data* $\mathbf{Y}$ with the stored profile $\mathbf{X}$ of the claimed user u, and returns 1 (accept), or 0 (reject). *Feature-based matching algorithms* [3,10] compare each feature in the profile against the corresponding feature in the claim, and produces a matching score that will be compared against a predefined threshold to accept or reject the claim. A BA system has (δ_1, δ_2)-*correctness*, if it satisfies $Pr[M(\mathbf{X}, \mathbf{Y}) = 0 \mid u = v] \leq \delta_1 \,\&\, Pr[M(\mathbf{X}, \mathbf{Y}) = 1 \mid u \neq v] \leq \delta_2$. False Acceptance Rate (FAR) and False Rejection Rate (FRR) are used to estimate the value of δ_1 and δ_2, respectively. One can combine two parameters into a single one, δ_E Equal Error Rate (EER), where FAR and FRR are equal. BA systems may use *vector-based* matching algorithm [8] too. For the BAVault, we need to use feature-based matching.

Attacks on BA Systems. Attacks on BA systems aim to fool the matching algorithm to accept the verification data of an invalid user. In a *mimicry attack* [25] the attacker attempts to mimic a user u by learning and mimicking u's behavior. An impersonation attack may use mimicry to claim a victim's identity. The success of this attack depends on the design of the BA system and the choice of features. We do not consider network-based attacks where the attacker eavesdrops communication of the valid user and uses their data as the verification data in an attacked session. Such attacks can be prevented by securing the communication channel using protocols such as Transport Layer Security (TLS) [18].

Random Projection (RP). RP is a distance preserving transformation that projects vectors in a high-dimensional space to a lower-dimensional space using a random projection matrix. The projection matrix $\mathbf{R}^{t\times d}, t < d$, projects a vector $\mathbf{x} \in \mathbb{R}^d$ to a vector $\mathbf{x'} = \mathbf{R}\mathbf{x}$, $\mathbf{x'} \in \mathbb{R}^t$, and (approximately) preserves the relative Euclidean distances between the vectors in the projected space. Elements of $\mathbf{R}$ are sampled from a standard normal distribution $N(0, 1)$ (see Lemma 1.3 of [23]). For faster computation we use discretized form of $N(0, 1)$ given by $Pr(x = +1) = \frac{1}{2\phi}$, $Pr(x = +0) = 1 - \frac{1}{\phi}$, $Pr(x = -1) = \frac{1}{2\phi}$. Distance preserving property of the resulting RP is shown in [1] for $\phi = 3$. It was shown in [20] that applying RP in the BA system will maintain correctness property of the system, and will also provide privacy for the profile.

Definition 1 (Fuzzy Vault [12]). *A fuzzy vault V_A is specified by a 4-tuple of parameters $(\mathbb{F}_q, r, t, k)$ and works as follows.*

1. *To encrypt a uniformly random secret message $m = (m_0 \cdots m_k) \in \mathbb{F}_q^{k+1}$, a polynomial $f(x)$ of degree k is constructed: $f(x) = \sum_{i=0}^{k} m_i x^i$.*
2. *The polynomial is then evaluated on a set of points $A = \{a_i \in \mathbb{F}_q : i = 1, \cdots, t\}$, called the* locking set. *The set $L = \{(a_i, f(a_i)) \in \mathbb{F}_q^2 : i = 1, \cdots, t\}$ forms the set of* legitimate points *in the vault.*
3. *To hide L, a random set of $r - t$* chaff points *$Ch = \{(a_i, \bar{a}_i) \in \mathbb{F}_q^2 : i = 1, .., r - t\}$ are selected with the property that $\bar{a}_i \neq f(a_i)$, to get spurious polynomials.*
4. *The fuzzy vault is obtained by permuting the elements of $V_A = L \cup Ch$, and is published together with the parameters $(\mathbb{F}_q, r, t, k)$.*
5. *To unlock V_A (decrypt the secret) using an* unlocking set *B, a decoding algorithm is used to find the best estimate of the legitimate points, which will be used to recover the message.*

The output of the decoding algorithm may contain legitimate and chaff points. To recover m, one can use Reed-Solomon (RS) decoder to correct the errors in the recovered set [12], or append a CRC to the secret message and use the pair $(m, CRC(m))$ in $f(x)$. The CRC will be used to identify the correct polynomial [17,21,22]. We will use this latter method that results in better error recovery.

Attacks on Fuzzy Vaults. Attacks can be grouped into, (i) algebraic attacks, and (ii) implementation attacks. In an algebraic attack, also called brute force attack, an attacker examines all candidate polynomials to find $f(x)$. In implementation attacks, the weaknesses of the vault implementation are used to recover the secret message. Lemma 1 of [6] gives the relation between the success probability of brute force attack and the number of spurious polynomials that the attacker must examine in biometric fuzzy vaults. In CRC-based biometric fuzzy vault, a brute force attacker must check on average $\binom{r}{k+1}\binom{t}{k+1}^{-1}$ polynomials [6]. Chang et al. [5] showed how to distinguish chaff points if they are not evenly distributed in vault space. Scheirer et al. [19] considered a number of attack scenarios including multiple vaults attack where the attacker knows multiple vaults (different m) with the same user biometric. This will allow the attacker to relate the data from different vaults to recover the secret. They also considered the loss of biometric privacy if the attacker learns the secret of a published vault.

3 BAVault

A BAVault uses a (δ_1, δ_2)-correct BA system (see Sect. 2) and profile data X of the BA system to lock the vault. The BAVault will open only by using the verification data Y of the BA system of the same user. The feature-based matching algorithm of the BA system employs a similarity function `Sim` $(.,.)$ to decide if two sets of samples have the same underlying distribution and output a confidence value $p_{i,j}$. A larger $p_{i,j}$ corresponds to higher confidence about the "sameness" of the distributions of the two sets. BAVault uses `Sim` $(.,.)$ function to measure the similarity of elements of B and V_A.

Definition 2. *A BAVault is defined by the tuple* $(\mathbb{F}_q, r, t, k, aux)$, *where the first four parameters are the same as Definition 1, and aux is the auxiliary data that can be provided with the vault. BAVault has a pair of algorithms* (VLock, VUnLock) *to lock and unlock the vault.*

1. VLock *works the same as Steps 1–4 in Definition 1 using the profile data* $\mathbf{X}$. *Locking algorithm uses the feature set of* $\mathbf{X}$ *to construct the* legitimate point set *for the secret value* $m \in \mathbb{F}_q^{k+1}$. *We use* A, L, *and* Ch *to denote the locking set, sets of legitimate points and chaff points, respectively. Points in* Ch *must each, (i) have a minimum "distance" from any legitimate point (to remain distinguishable by the BA matching algorithm), and (ii) should not satisfy the polynomial* $f(x)$.
2. VUnLock *has a* soft-decision decoding algorithm `CGen` *which uses the* `Sim` $(.,.)$ *function of the BA system to construct a* $r \times t$ *confidence matrix* $[p_{i,j}]$, *that will be used to recover the* $f(x)$.

(Δ_1, Δ_2)**-correctness:** Consider a (δ_1, δ_2)-BA system that uses a feature matching function `Sim` $(.,.)$, a matching algorithm $M(.,.)$. Let $\mathbf{X}$ and $\mathbf{Y}$ be two BA profiles (or claims). For a uniformly distributed secret m, a BAVault $V_A \leftarrow$ VLock$(\mathbf{X}, m)$ is (Δ_1, Δ_2)-correct if the conditions: (i) $Pr[\text{VUnLock}(V_A, \mathbf{Y}) \neq m | M(\mathbf{X}, \mathbf{Y}) = 1] \leq \Delta_1$, and (ii) $Pr[\text{VUnLock}(V_A, \mathbf{Y}) = m | M(\mathbf{X}, \mathbf{Y}) = 0] \leq \Delta_2$ hold. The correctness property[1] of the BAVault is also referred to as Δ_1-FRR and Δ_2-FAR.

Security. We evaluate the security of a BAVault in (a) protecting the secret m, and (b) ensuring the privacy of X. Following Kirckoff's principle, we assume the details of the vault system is public. For (a) we consider a number of attacks, including when the attacker has access to multiple vaults. To evaluate (b), we assume the attacker knows a vault V_A and its corresponding message m.

Attacker strategies. For (a), that is message recovery, we consider (i) algebraic attacks, and (ii) system (implementation) attacks, including attacks that exploit non-uniform distribution of vault points, attack using multiple vaults of the same user, and using impersonation attack on the BA system. For (b), that is profile privacy, the attacker knows the secret message of a published vault that allows them to find the points in the vault that correspond to true features.

[1] We used FAR and FRR to evaluate BAVault correctness.

4 A Secure BAVault

Let the locking and unlocking sets be of the same size, that is $|A| = |B|$. Figure 1a and 1b are the block diagrams of the BAVault locking and unlocking algorithms. They are shown in the Appendix, Algorithm 1 and Algorithm 2.

4.1 Locking

The locking algorithm takes m and $\mathbf{X}$ and does the following.

Message m. The message is appended with a CRC (preprocessing) and is used to construct the polynomial $f(x)$.

Profile Data. The profile $\mathbf{X}$ will be transformed to $\mathbf{X}' = \mathbf{RX}$ by using an RP (preprocessing). Because of the distance-preserving property of RP, the relative distances among profile vectors will be maintained and so the correctness of the feature matching algorithm will not be affected. Assuming that the distribution of the features F_i in the profile are normal[2], the distribution of the projected features F_i' will also be normal. We thus represent F_i' with its mean and variance, $a_i = (\mu_i, \sigma_i^2)$. To map (μ_i, σ_i^2) to an element of $\mathbb{F}_q$ (locking set generation), we write μ_i and σ_i^2 as binary strings, and concatenate them. The number of binary digits for each component is determined by the required correctness parameters (Δ_1, Δ_2), and will determine the finite field size. The set A consists of $\mathbb{F}_q$ values of all transformed features. The mapping of (μ_i, σ_i^2) to an element of $\mathbb{F}_q$ is one-to-one and invertible. The polynomial $f(x)$ is then evaluated on elements of $A = \{a_i\}_{i=1}^t$ to form the *legitimate points set* $L = \{(a_i, f(a_i)) \in \mathbb{F}_q^2\}_{i=1}^t$.
Generating chaff point needs feature similarity evaluation. A *feature similarity function* $\mathtt{Sim}^{\mathtt{MV}}(.,.)$ measures the closeness (sameness) of two distributions whose estimated[3] means and variances are $a_i = (\mu_i, \sigma_i^2)$ and $b_j = (\mu_j, \sigma_j^2)$, respectively. The function uses (i) `tTest`- test for the sameness of μ_i and μ_j, and (ii) `fTest`- test for the sameness of σ_i^2 and σ_j^2, to obtain two confidence values $p_{i,j}^m$ and $p_{i,j}^v$, respectively, and combines them by a metadata analyzer `FMethod` [4] (Fisher's method) to obtain a final confidence value.

Available Chaff Points. Chaff points of the form $(a_i, \bar{a_i})$ must be chosen such that (i) a_i does not correspond to the parameters of a distribution that is close to the distributions of the true features, and (ii) $\bar{a_i} \neq f(a_i)$. For given ranges R_1, R_2 of mean and variance, each a subinterval of $\mathbb{R}$, we first find D_F the number of distinct variances in R_2 that can be distinguished using `fTest` test. For this, we start with the lowest value of the variance R_2, σ_0^2, and find smallest σ_1^2 for which $\mathtt{Sim}^{\mathtt{MV}}(a_0, a_1) = 0$ for a chosen significance value α. That is the variance test will consider σ_1^2 not similar to σ_0^2. We repeat this starting from σ_1^2, and continue until the next σ_i^2 is outside the upper limit of the range R_2. For each found value of σ_i^2, we find $D_{T,i}$, the number of distinguishable means using a similar approach (now using `tTest`) within the range R_1. This gives an estimate of the number of available X-coordinates for chaff points.

[2] This is true for features of BA systems that have been used in our experiments.
[3] From the corresponding sample sets.

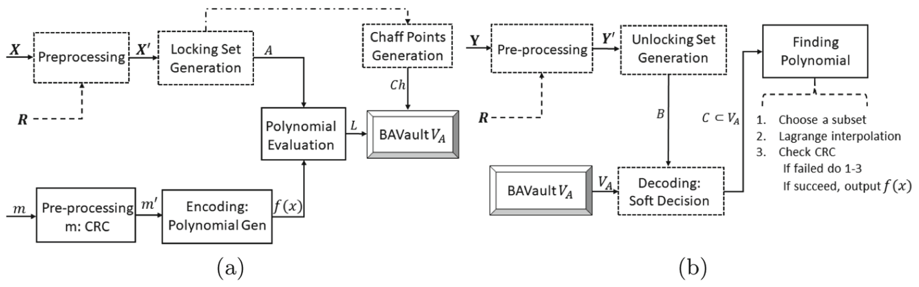

Fig. 1. Block diagram of BAVault (a) locking and, (b) unlocking.

Lemma 1. *The available X-coordinate for chaff points is upper bounded by* $Q_X = \sum_{\sigma_i^2 \in D_F} |D_{T,i}|$.

Generating Chaff Points. To provide even distribution for vault points, we use algorithm `ChaffGen` (see Algorithm 3 in Appendix) for chaff point generation. It divides $\mathbb{F}_q^2$ into ν^2 subareas and attempts to put almost the same number of vault points in each subarea. The algorithm generates random chaff points $(a_i, \bar{a}_i)$ (that satisfy, a_i distinguishable from all feature values, and $\bar{a}_i \neq f(a_i)$), and place them in ν if they satisfy the bound on corresponding subarea, and reject otherwise. The algorithm first obtains an estimate of the vault size r by taking into account the required security against brute force attack, determines r/ν^2, the number of allowable points in a subarea. It then obtains the number of chaff points in a subarea by subtracting the number of true feature points in a subarea from r/ν^2. To evaluate `ChaffGen`, we used Kolmogorov-Smirnov (KS) test [11] to estimate the uniformity of vault points, and also the distribution of their X-coordinates. The vault V_A is obtained by permuting the points in each subarea.

4.2 Unlocking

The unlocking algorithm takes a V_A and the verification data $\mathbf{Y}$ of a user as input. It uses the published helper data (aux) to generate $\mathbf{R}$ that is used to transform $\mathbf{Y}$, resulting in the unlocking set B.

Soft Decision Decoding `CGen`. For an unlocking set B, we find a matching subset of V_A in two steps: (i) a feature matching algorithm `FMatch`, and (ii) a set matching algorithm `SMatch`. Feature matching `FMatch` uses $\texttt{Sim}^{\texttt{MV}}(.,.)$ to construct an $r \times t$ matrix $\texttt{ConF} = [P_{i,j}]$ of confidence values, by comparing each element of B against each element of V_A. The set matching algorithm finds a subset of V_A that is the *best match*. We define the best matching subset of the vault points as a subset that maximizes the total confidence value. The set matching `SMatch` must find a subset of r columns of `ConF`, and in each column chooses exactly one element, such that no two elements are in the same row. We formulate this as a Linear Assignment Problem (LAP) [2] by augmenting `ConF` to $\hat{\texttt{ConF}}$, a $r \times r$ square matrix with zeros in all new entries. LAP is a fundamental combinatorial optimization problem that minimizes the total cost of assigning agents to tasks when all agent-task pairs are possible but have different costs. There are efficient algorithms for solving the LAP. The output of `CGen` is a set C (see Algorithm 4 in Appendix).

Theorem 1. *Algorithm* `CGen` *outputs a set C that has the highest total confidence of matching B, in $O(r^3)$ number of steps.*

The final step is the polynomial reconstruction from C, and uses the CRC of the secret message to recover the correct polynomial.

4.3 Security Analysis

Message recovery by brute force attack needs searching among at least $\frac{\psi}{3}q^{k-t}(r/t)^t$ spurious polynomials for every $\psi > 0$ with probability $1 - \psi$ (see Lemma 1 of [6]) that go through t vault points, or $\binom{r}{k+1}\binom{t}{k+1}^{-1}$ polynomials that go through $k + 1$ vault points. The chaff points generation algorithm will ensure the chaff points are evenly distributed and cannot be distinguished from true feature points.

Multi-vault Attack. Consider two vaults V_A^i and V_A^j for a profile $\mathbf{X}$. Using two different random matrices $\mathbf{R}_i$ and $\mathbf{R}_j$ ensures that the projected profiles $\mathbf{X}'_i = \mathbf{R}_i\mathbf{X}$ and $\mathbf{X}'_j = \mathbf{R}_j\mathbf{X}$ are independent and will not leak any information about $\mathbf{X}$.
Impersonation (mimicry) attack on the underlying BA system will also break the BAVault. This is similar to the attack on the biometric-based vault. A careful selection of the behavioral features will protect against this attack.
Profile privacy requires profile data to be protected even if the attacker knows the secret message of a vault. Using m and the vault, an attacker will only be able to recover the projected profile $\mathbf{X}'$. However, as shown in [20], recovering $\mathbf{X}$ from $\mathbf{X}'$ using the *minimum-norm* solution to $\mathbf{R}$, which is the best-known estimator of $\mathbf{X}$ from $\mathbf{X}'$, cannot recover the original $\mathbf{X}$.

5 BAVault Implementation

We implemented and evaluated our proposed BAVault using Touchalytics [8] and DAC [10] data.

Touchalytics uses users' touch data (up-down and left-right scrolling) when interacting with an app, and uses a vector-based matching algorithm to achieve an EER of less than 3.0%. The system uses 30 behavioral features and data from 41 users.

DAC uses the behavioral features of users that are collected while drawing random challenge circles that are presented to them, to verify their verification claims. We extended DAC [10] and added a new set of features. This results in 65 features and reduces the EER of DAC from 5.0% to 1.05%.

Experiment Setup. We downloaded and cleaned[4] Touchalytics data before using them. There are 41 profiles and 41 valid verification claims. For DAC, we collected data from 199 Amazon Mechanical Turks (AMT). Suitability of AMT for cognitive behavioral experiments has been confirmed in [7]. After removing outlier data (around 2.66%), we obtained 195 profiles and 891 valid verification claims (each Turk had multiple verification attempts).

To obtain reliable features distributions from the collected data, we combined all user data, shuffled them, and divided them into two halves: the first half was used for vault locking and the second half was used for vault unlocking. Every profile (unlocking claim) has 93–615 vectors of dimension 30 in Touchalytics BAVault, and 40–120 vectors of dimension 65 in DAC BAVault. Against each vault, there were one valid unlock attempt and 5 invalids unlock attempts from 5 randomly chosen users. The locking and unlocking of Touchalytics and DAC based BAVault takes (10.78 and 1.66 seconds) and (26.89 and 3.37 seconds), respectively, on a desktop that uses Intel Core(TM)i5-2400 CPU (3.10 GHz), 8 GB RAM.

Feature Encoding. For RP, the dimension of projected spaces for Touchalytics and DAC are $t = 25$ and $t = 45$, respectively. We generated $\mathbf{R}$ from the discrete distribution uses in [1] and normalized profile data after RP. To measure distinctiveness and normality of BA features we used $\mathtt{Sim}^{\mathtt{MV}}(.,.)$ function, and Chi-square goodness-of-fit test. Touchalytics and DAC profiles had 97.37% and 92.81% distinct features before RP, and 99.82% and 97.73%, after RP, respectively. The normality test results for the two systems before and after RP are 2.32% and 53.4%, and 65.56%, 54.83%, respectively. EER of both BA systems after RP remained almost the same; 1.20% in Touchalytics and 4.66% in DAC.

To encode (μ, σ^2) as an element of $\mathbb{F}_q$, we remove the decimal points of μ_i and σ_i^2, take the three most significant digits of each, and concatenate them. For σ^2 we only consider two digits because the first digit after the decimal place is always zero. In Touchalytics profiles all $a_i \in A$ are in the range [15400, 84600], and in DAC profiles they are in [15000, 80500]. We bring all the data to the range [0,65535] by subtracting the lowest value, and cutting off all the values above 65535[5] and then represented them as a binary string in $\mathbb{F}_{2^{16}}$.

Generate Chaff Points. We estimated Q_X for both BAVaults. The mean and variance range for the features in Touchalytics is between [230.0, 820.0] and [1.0^2, 52.0^2], respectively. For DAC, the corresponding values are [108.0, 875.0] and [1.0^2, 98.0^2], respectively. For $N_i = N_j = 300$ (average samples in a features) in Touchalytics BAVault, D_F allows 24 distinct σ^2, and for each $\sigma_i^2 \in D_F$ the size of all D_T are $1896, 884, \cdots, 42$, respectively. In DAC BAVault for $N_i = N_j = 80$ the set size $|D_F| = 21$ and all $|D_T|$ are $2464, 1039, \cdots, 33$, respectively.

To distribute vault points evenly, we divided the range of both X and Y-axis of both vaults into equal size segments which produce $\nu^2 = 25$ subareas. We

[4] We replace 'NaN' and 'Infinity' by zero and dropped the 'doc id', 'phone id', and 'change of finger orientation' columns.

[5] In Touchalytics there are around 3.70% of a_i that are out of the range [0, 65535] and in DAC it is only 0.32%. This rounding slightly affects BAVault correctness.

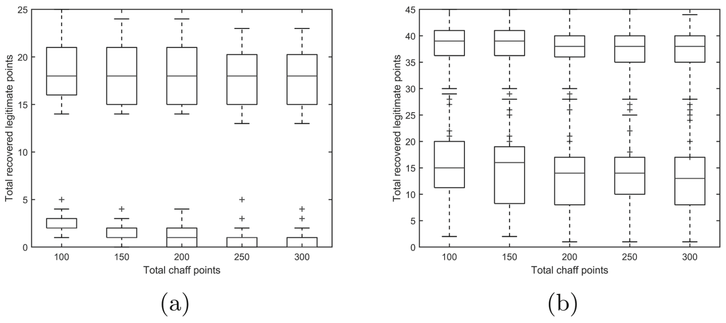

Fig. 2. The distribution of recovered legitimate points for both valid and invalid claims for the different number of chaff points. Figure (a) for Touchalytics based BAVault and (b) for DAC based BAVault.

chose $|V|$ =125–325 for Touchalytics and $|V|$ =145–345 for DAC based BAVault. `ChaffGen` algorithm counts the number of legitimate points in each subarea and added random points to each subarea when possible, taking into account the total vault size $|V|$. The KS-test gives average confidence value for the uniformity of Touchalytics and DAC based BAVault as 0.69 and 0.66, respectively. The average confidence values of KS-test for uniformity of X-components of the two vaults are 0.89 and 0.78, respectively.

Recovering Legitimate Points. The `CGen` algorithm of the BAVault outputs a set $C \subset V_A$ that has $\hat{t} \leq t$ legitimate points out of t recovered points. The value of $\hat{t}$ depends on the unlocking claim. In our experiments, we added 100–300 chaff points to each vault. This number can be increased at the cost of increased encoding and decoding time. Figure 2 is the recovered legitimate points in a Box-plot for both valid and invalid claims. The valid and invalid claims can recover 15–22 and 0–3 (Touchalytics) and 33–41 and 7–20 (DAC), legitimate points. A valid user may not be able to recover all legitimate points because of the variability of the user's behavior, and an attacker may be able to recover some of the legitimate points of a target vault by using attacker's profile and public data $\mathbf{R}$. The Box-plots show that the gap between the first quartile corresponding to the valid claims and the third quartile corresponding to the invalid claims is large and both BAVaults work correctly. The gap increases with the number of chaff points.

BAVault Correctness and Security. Table 1 summarizes correctness and security of the two BAVaults. The values are inline with existing fuzzy vault systems. The degree of the polynomials in Touchalytics and DAC based BAVaults are 12–13, and 28–30 respectively, resulting in FAR and FRR to be 0.0% and 2.43%, and 2.56% and 4.65%, respectively. The secret sizes in the two cases are 192–208 bits, and 448–480 bits, respectively. For a valid polynomial that goes through t or $k + 1$ valid vault points, the brute force attacker will

Table 1. Both vaults ensure sufficient correctness and security. For its higher number of features, DAC based BAVault allows larger secret size than Touchalytics based BAVault.

	Touchalytics		DAC	
FAR	degree of $f(x)$: k = 12-13	0.0%	degree of $f(x)$: k = 28-30	0.0%-2.56%
FRR		0.0%-2.43%		3.89%-4.65%
Size of m (bits)		192-208		448-480
$f_{cand}(x)$(CRC)		$2^{54} - 2^{58}$		$2^{101} - 2^{110}$
Spurious polynomials		2^{20}-2^{26}		$2^{41} - 2^{56}$

need to check 2^{20}-2^{26} and 2^{41}-2^{56} spurious polynomials, or 2^{54}-2^{58} and 2^{101}-2^{110} candidate polynomials in both BAVaults, respectively.

Multi-vault security. Two vaults of a user will have two different vault point sets. To investigate possible residual relation between the two vaults that share true features, we used a modification of `CGen` algorithm which takes V_A^i and V_A^j and matches each X-element of the first set against all X-elements of the second set, and returns a subset C that has the highest total confidence value. This recovers only 3.0% and 7.0% legitimate points in Touchalytics and DAC based BAVault, respectively.

Protection Against Impersonation Attack. We considered a pair of profiles $\mathbf{X}_i$ and $\mathbf{X}_j$, that have 5.0%–16.0% overlapping features (e.g. from a mimicry attack). We then used $\mathbf{X}_i$ to construct a BAVaultand used $\mathbf{X}_j$ to recover the legitimate points from the vault. This can recover around 2.0%–9.0% more legitimate points compared to an invalid claim. This is, however, not sufficient to open the vault.

6 Concluding Remarks

BAVault offers significant advantages over biometric-based fuzzy vaults. We outlined challenges of implementing an efficient and secure BAVault, proposed a design that addresses these challenges, and validated our design analytically and experimentally. Our work can be extended to use higher-order statistics for representing features. Another direction is to employ *ranked assignment problem* to use the top t highest ranking sets to improve reliability.

A Appendix

Algorithm 1: $V_A \leftarrow \mathbf{VLock}(\mathbf{X}, m)$

```
1: Construct a polynomial. For a secret message $m$, appends CRC to $m$ to obtain $m' = [m_0, m_1, \cdots, m_k] \in \mathbb{F}_q^{k+1}$; define $f(x) = \sum_{i=0}^{k} m_i x^i$.
2: Profile projection. Use RP to transform $\mathbf{X} \in \mathbb{R}^{n\times d}$ to $\mathbf{X}' \in \mathbb{R}^{n\times t}$ $(t \leq d)$. The random seed that is used to generate $\mathbf{R}$ is the helper data (aux).
3: Locking set generation. Each feature $F_i' \subset \mathbf{X}'$ will be represented by its mean and variance, $(\mu_i, \sigma_i^2)$, and encoded to $a_i \in \mathbb{F}_q$ to form a set $A = \{a_i\}_{i=1}^t$.
4: Polynomial evaluation. The polynomial $f(x)$ is evaluated on the elements of $A$ to obtain the set of legitimate points $L = \{(a_i, f(a_i)) \in \mathbb{F}_q^2\}_{i=1}^t$.
5: Chaff point generation. Generate $Ch = \{(a_i, \bar{a}_i) \in \mathbb{F}_q^2, i = 1, \cdots, r-t, \bar{a}_i \neq f(a_i)\}$ by using ChaffGen algorithm, taking into account $t$ points in $L$, and polynomial $f(x)$. Chaff points must satisfy the required properties.
6: The vault. Permute the elements of $V_A = L \cup Ch$ to obtain $V_A$.
```

Algorithm 2: $\{m, \perp\} \leftarrow \text{VUnLock}(V_A, \mathbf{Y})$

```
1: Claim projection. The claim $\mathbf{Y}$ will be transformed to $\mathbf{Y}'$, using $\mathbf{R}$ that can be reconstructed using aux.
2: Unlocking set generation. Each sample set $F_j' \in \mathbf{Y}'$ will be summarized to a pair $b_j = (\mu_j, \sigma_j^2) \in \mathbb{R}^2$; the set of $t$ pairs will form the set $B = \{b_j\}_{j=1}^t$.
3: Recovering the legitimate points. The soft-decision decoding algorithm CGen recovers the legitimate points from $V_A$. The algorithm CGen$(V_A, B)$ has two steps: Step 1: FMatch uses Sim$^{\texttt{MV}}$(.,.) for each pair of elements of $B$ and $V_A$; Step 2: SMatch uses an optimization algorithm to find the "best" matching subset $C \subset V_A$.
4: Recover the secret. A candidate polynomial $f_{cand}(x) = \sum_{i=0}^{k} m_i^* x^i$ of degree $k$ is constructed from $k+1$ points of $C$. If the coefficients of $f_{cand}(x)$ do not satisfy the CRC, $f_{cand}(x)$ is rejected, and a new set is chosen. The process will be repeated until $m$ is found, or $\perp$ is outputted, indicating no polynomial was found.
```

Algorithm 3: $V_A \leftarrow \texttt{ChaffGen}(\mathbb{F}_q, L, r)$

```
INPUT/OUTPUT:
    $\mathbb{F}_q^2$: Vault space
    $L$: Legitimate points set
    $r$: Total vault points
    $V_A$: A set of $r$ points.
 1: Divide $\mathbb{F}_q^2$ in $\nu^2$ subareas
 2: $|\nu_{max}| = \lceil \frac{r}{\nu^2} \rceil$
        ▷ max allowable-points in a subarea
 3: for each $\nu_i$ do
 4:     $|\nu_i| \leftarrow$ number of $(a_i, f(a_i)) \in L$
 5:     if $|\nu_i| < |\nu_{max}|$ then
 6:         $w(\nu_i) = [\frac{1}{\nu^2}(1 - \frac{|\nu_i|}{|\nu_{max}|})]$
                                    ▷ calculate the weight
 7:     else
 8:         $w(\nu_i) = 0$
 9:     end if
10: end for
11: for each $k \le r - |L|$ do
                                    ▷ for each chaff point
12:     pick a $\nu_i$ based on $w(\nu_i)$
13:     choose $a_i$ for $\nu_i$ from $X$-axis of $V_A$
            where $\forall b_j \in V_A$, $\texttt{Sim}^{\texttt{MV}}(a_i, b_j) = 0$
                        ▷ $X$-component of the chaff point
14:     choose $\bar{a}_i$ for $\nu_i$ from $Y$-axis of $V_A$
            where $\bar{a}_i \ne f(a_i)$
                        ▷ $Y$-component of the chaff point
15:     $Ch \leftarrow \{(a_i, \bar{a}_i)\}$
                            ▷ add chaff point in $Ch$
16:     Update $V_A \leftarrow L \cup Ch$
17:     $|\nu_i| \leftarrow |\nu_i| + 1$
18:     $w(\nu_i) = [\frac{1}{\nu^2}(1 - \frac{|\nu_i|}{|\nu_{max}|})]$
                                    ▷ update weight
19: end for
20: return $V_A$
```

Algorithm 4: $C \leftarrow \texttt{CGen}(V_A, B)$

```
INPUT/ OUTPUT:
    $V_A = \{(a_1, a_1^*), .., (a_r, a_r^*)\}$
    $B = \{b_1, b_2, \cdots, b_t\}$
    $C \subset V_A$: $t$ pairs of points
 1: $\texttt{ConF} = \texttt{FMatch}(V_A, B)$
 2: $\Phi = \texttt{SMatch}(\texttt{ConF})$
 3: for each $i \le r$ do
 4:     for each $j \le t$ do
 5:         if $\Phi[i, j] = 1$ then
 6:             $C \leftarrow (a_i, a_i^*)$
 7:         end if
 8:     end for
 9: end for
10: return $C$
    //Pseudocode of FMatch
11: move $\forall a_i \in \mathbb{F}_q$ to $a_i = (\mu_i, \sigma_i^2) \in \mathbb{R}^2$
12: for each $a_i \in V_A$ do
13:     for each $b_j \in B$ do
14:         $p_{i,j}^m \leftarrow \texttt{tTest}(\mu_i, \mu_j)$;
                $\mu_i \in a_i$ and $\mu_j \in b_j$
15:         $p_{i,j}^v \leftarrow \texttt{fTest}(\sigma_i^2, \sigma_j^2)$;
                $\sigma_i^2 \in a_i$ and $\sigma_j^2 \in b_j$
16:         $p_{i,j} \leftarrow \texttt{FMethod}(p_{i,j}^m, p_{i,j}^v)$
17:         $\texttt{ConF}[i, j] \leftarrow p_{i,j}$
18:     end for
19: end for
20: return ConF
    //Pseudocode of SMatch
21: add $r - t$ pseudo points
    $\hat{B} = \{b_1, b_2, \cdots, b_t, \hat{b}_{t+1}, \cdots, \hat{b}_r\}$
22: for each $i \le r$ do
23:     for each $j \le r$ do
24:         if $j > t$ then
25:         $\overline{\texttt{ConF}}[i, j] \leftarrow 0$
26:         else
27:     $\overline{\texttt{ConF}}[i, j] \leftarrow \texttt{ConF}[i, j]$
28:         end if
29:     end for
30: end for
31: $\Phi \leftarrow \texttt{LAP}(\overline{\texttt{ConF}})$
            ▷ assign 1 for optimal subset or 0
32: return $\mathbf{\Phi}$
```

References

1. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. JCSS **66**(4), 671–687 (2003)
2. Akgül, M.: The linear assignment problem. In: Akgül, M., Hamacher, H.W., Tüfekçi, S. (eds.) Combinatorial Optimization. NATO ASI Series (Series F: Computer and Systems Sciences), vol. 82, pp. 85–122. Springer, Heidelberg (1992). https://doi.org/10.1007/978-3-642-77489-8_5
3. Alimomeni, M., Safavi-Naini, R.: How *to prevent* to delegate authentication. In: Thuraisingham, B., Wang, X.F., Yegneswaran, V. (eds.) SecureComm 2015. LNICST, vol. 164, pp. 477–499. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28865-9_26
4. Brown, M.B.: A method for combining non-independent, one-sided tests of significance. Biometrics **31**, 987–992 (1975)
5. Chang, E.C., et al.: Finding the original point set hidden among chaff. In: Proceedings of the ASIACCS 2006, pp. 182–188. ACM (2006)
6. Clancy, T.C., et al.: Secure smartcardbased fingerprint authentication. In: Proceedings of the ACM SIGMM WBMA 2003, pp. 45–52. ACM (2003)
7. Crump, M.J., et al.: Evaluating Amazon's mechanical turk as a tool for experimental behavioral research. PLoS One **8**(3), e57410 (2013)
8. Frank, M., et al.: Touchalytics: on the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE Trans. Inf. Forensics Secur. **8**(1), 136–148 (2013)
9. Hahs-Vaughn, D.L., et al.: Statistical Concepts: A Second Course. Routledge, Abingdon (2013)
10. Islam, M.M., et al.: Poster: a behavioural authentication system for mobile users. In: Proceedings of the ACM CCS 2016, pp. 1742–1744. ACM (2016)
11. Massey, Jr., Frank, J., et al.: The Kolmogorov-Smirnov test for goodness of fit. JASA **46**(253), 68–78 (1951)
12. Juels, A., et al.: A fuzzy vault scheme. Des. Codes Crypt. **38**(2), 237–257 (2006). https://doi.org/10.1007/s10623-005-6343-z
13. Kaur, M., et al.: Fuzzy vault template protection for multimodal biometric system. In: Proceedings of the ICCCA 2017, pp. 1131–1135. IEEE (2017)
14. Lee, Y.J., Bae, K., Lee, S.J., Park, K.R., Kim, J.: Biometric key binding: fuzzy vault based on iris images. In: Lee, S.-W., Li, S.Z. (eds.) ICB 2007. LNCS, vol. 4642, pp. 800–808. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74549-5_84
15. Li, C., et al.: A security-enhanced alignment-free fuzzy vault-based fingerprint cryptosystem using pair-polar minutiae structures. IEEE Trans. Inf. Forensics Secur. **11**(3), 543–555 (2015)
16. Mankiewicz, R.: The Story of Mathematics. Cassell, London (2000)
17. Nandakumar, K., et al.: Fingerprint-based fuzzy vault: implementation and performance. IEEE TIFS **2**(4), 744–757 (2007)
18. Salowey, J.A., et al.: TLS 1.3, 10 August 2018. https://www.ietf.org/blog/tls13/. Accessed 22 Apr 2019
19. Scheirer, W.J., et al.: Cracking fuzzy vaults and biometric encryption. In: Biometrics Symposium 2007, pp. 1–6. IEEE (2007)
20. Taheri, S., Islam, M.M., Safavi-Naini, R.: Privacy-enhanced profile-based authentication using sparse random projection. In: De Capitani di Vimercati, S., Martinelli, F. (eds.) SEC 2017. IAICT, vol. 502, pp. 474–490. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58469-0_32

21. Uludag, U., Pankanti, S., Jain, A.K.: Fuzzy vault for fingerprints. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 310–319. Springer, Heidelberg (2005). https://doi.org/10.1007/11527923_32
22. Uludag, U., et al.: Securing fingerprint template: fuzzy vault with helper data. In: Proceedings of the CVPRW 2006. pp, 163–163. IEEE (2006)
23. Vempala, S.S.: The Random Projection Method, vol. 65. American Mathematical Society, Providence (2005)
24. Wu, L., et al.: A face based fuzzy vault scheme for secure online authentication. In: Proceedings of the ISDPE 2010, pp. 45–49. IEEE (2010)
25. Yampolskiy, R.V.: Mimicry attack on strategy-based behavioral biometric. In: Proceedings of the ITNG 2008, pp. 916–921. IEEE (2008)