Secure remote matching with privacy: Scrambled Support Vector Vaulted Verification (S^2V^3)

Michael J. Wilber and Terrance E. Boult
Vision and Security Technology Lab, UCCS
Colorado Springs, CO, 80918, USA
{mwilber,tboult}@vast.uccs.edu

Abstract

As biometric authentication systems become common in everyday use, researchers are beginning to address privacy issues in biometric recognition. With the growing use of mobile devices, it is important to develop approaches that support remote mobile verification. This paper outlines the need for a mobile/remote SVM-based authentication system that does not compromise the privacy of the subject being recognized. We discuss limitations of earlier privacy-preserving authentication systems and present necessary privacy and security requirements that make a system attractive from both the server's security point of view and from the client's privacy-centric point of view. We then present a novel protocol we call "Vaulted Verification" that allows a server to remotely authenticate a client's biometric in a privacy preserving way. We conclude with a small evaluation of performance, discussion of security implications, and ideas for future work.

1 Introduction

The underlying question addressed in this paper is "Can we have a secure, privacy-preserving means of remote vision-based verification using support vector machines?" The answer, of course, is yes. This paper presents a novel approach to privacy-enhanced verification. We aim to solve the important problem of improving recognition rates by allowing SVM-based models while at the same time allowing remote verification to improve privacy and provide secure verification of the match. While the core algorithm can be applied to other recognition problems, this paper will focus on face-based biometric verification. This paper lies at the intersection of four areas of vision-related research: support vector machines for recognition, remote and mobile authentication, face recognition, and privacy enhanced recognition.

The problem that motivates our research is the ability to perform remote verification. Suppose a client, say Chris (C), wishes to use a mobile phone or laptop to remotely authenticate to a central server, say a bank (B). Both parties require the system to be secure and Chris wishes to maintain his privacy. With classical solutions, Chris enrolls and the bank maintains a central database with his biometric template and biographic data. When Chris wants to access his bank account, he provides his biometric and biographic data for verification, and the bank separately matches these items. This may seem secure for the bank, but Chris has privacy and security concerns about the server storing his enrollment record, receiving his raw biometric data, and being able to match on his raw data whenever desired. How can Chris ensure the data sent for verification is not being phished? How can Chris ensure no one uses his recognition data without his consent?

One solution to this privacy/security conundrum is to use privacy-enhanced template protection(PETP). Earlier techniques such as cancelable biometrics [21] and fuzzy vaults [12] have been explored but have significant security problems [23, 18]. Some PETP techniques such as revocable biotokens [4] and fuzzy commitment schemes [13] are still secure but are not suitable for remote matching.[1]

Other approaches from the domain of data mining involve multiplying feature vectors by a random orthogonal matrix to "rotationally perturb" each vector [6]. Unfortunately, attacks based on Independent Component Analysis can compromise those methods' security [10, 17]. A similar method that discards information is called "random projection," [3, 17] originally intended for dimensionality reduction but recently applied to privacy-preserving biometric systems [24, 14, 9, 2, 11]. These random-projection oriented approaches provide some protection but they have security limitations and weak performance.

Because the client values privacy, ideally the server should *never* have access to biometrically identifiable data;

[1] While [4] presents high accuracy claims, it has since been shown there were problems in their experimental protocol, such that testing data was inappropriately leaked into their "robust windows" sizing. When properly separated, the performance of robust revocable face biotokens are only slightly better than classic PCA.

not at enrollment, not at matching, and definitely not stored on the server. However, new problems arise if the server no longer controls the matching. With a traditional biometric system or with traditional privacy-enhanced technologies, there is no way for the server to verify the remote match. Thus, an important goal of this paper is to develop a method that supports secure remote matching.

In the field of non-privacy-preserving face recognition, using multi-view galleries for training is now a widely-accepted technique to improve performance. Many state of the art systems use SVM classifiers, e.g. [20, 15]. This raises an interesting question: How can we combine multiple views and use general SVM classifiers in a privacy-preserving setting?

To meet these goals, we show how to build a privacy-preserving SVM-based verification system. With our system, a server can use an SVM classifier to verify a client's authenticity by stepping through a challenge-response protocol. This is done in such a way that the server cannot learn anything about the client from the trained SVM template. Furthermore, classification can only commence with the client's cooperation, making "automatic" classification impossible. These two properties help preserve the client's privacy. To show our method's feasibility, we implement a remote face authentication system.

2 Privacy and security requirements

What does it mean for a classification method to be "private"? For our purposes, we define a "private" verification system as one that is resistant to the following attacks [23, 7, 8]:

- **Man-in-the-middle or replay attacks**: During authentication, eavesdroppers/attackers must not be able to capture any useful information, nor should they obtain information that describes the subject's biometric.
- **Record multiplicity attacks**: Templates of the same subject across multiple databases must not be linkable.
- **Compromised template attacks**: If an attacker acquires the template on the server, they must not be able to learn anything unique about the subject's biometric, and they must not be able to authenticate as the subject.
- **Automated identification attacks**: Classification must only be possible with the subject's consent. Verification without the subject's knowledge (e.g. in an automatic terrorist-identification system) must be impossible.
- **Blended substitution attacks**: A malicious insider (with all server keys) must not be able to *silently* replace the subject's template with one that matches the subject and the attacker. Doing so would allow the attacker to "backdoor" the authentication system. [23]
- **Non-revocability concerns**: The classifier template should be "revocable"; the subject should be able to destroy the template simply by destroying a passphrase used in the authentication process.

Finally, all these requirements should be achievable without a large drop in verification accuracy.

Our SVM-based verification system is novel because it protects against the six attacks outlined above while still providing useful accuracy for many verification problems. To our knowledge, no other SVM-based system exists that satisfies these requirements.

3 Previous work in private SVMs

There is little research exploring privacy-preserving SVM classification. Certain methods exist, but they only focus on protecting the training set, not the final classifier.

Two methods exist for allowing multiple clients with different shards of the training set to construct an SVM classifier without revealing their portion of the set to the other participants [25, 26]. Neither method addresses protecting the SVM itself after training; the final classifier is assumed to be securely stored by a trusted third party.

Another method describes a way of post-processing a trained SVM to protect privacy of the support vectors [16], but this approach only works on Gaussian kernels because it discards terms of the Gaussian function. In a sense, the final SVM produced by this method trades privacy for accuracy. The final SVM may still leak information about what types of vectors it classifies.

All three methods focused on problems such as medical classifiers where the goal was to release the final classifier while still protecting the privacy of individual patients' support vectors. The final SVM was left slightly altered [16] or unprotected [25, 26]. If an attacker acquired a copy of the SVM classifier, they could acquire the support vectors that store the subject's identification and could feasibly reconstruct the subject's biometric. Thus, these schemes are unsuitable for our problem. Ideally, we wish to build an SVM matching system that can classify samples without revealing anything about the samples it classifies.

4 Remote Authentication Issues

For security, the server will not just trust a remote client responding yes/no on a match. To resist automatic authentication attacks, the server cannot store enough information to authenticate the client without the client's knowledge. The ideal solution is to have the client and server step through a challenge-response protocol for each authentication.

We wish to build a protocol that satisfies the privacy requirements outlined in Section 2. To do this, several key questions must be carefully considered. Which party matches the template against the probe? How is the template stored? In a traditional protocol, if the server matches the client's biometric against a stored template, it is the server's responsibility to keep the template and the matching data secret. Further, if an eavesdropper captures the biometric or the stored template, they can compromise the privacy of the subject. On the other hand, if the client matches the biometric against the template, the server must trust the client's response because the match score alone has no inherent security value. In such a protocol, the server has no way of double-checking the client's match claim.

This creates a conflict. For security, the server must match the biometric because it cannot trust the client. For privacy, the client must match the biometric because if the server gets the raw data, it could compromise the client's privacy.

5 SVM Vaulted Verification Protocol

To satisfy the privacy requirements outlined in Section 2, we present a full authentication protocol that allows a server to use SVM classifiers to authenticate a client in a privacy-preserving way. This is presented for secure face verification but should be useful for other modalities/domains.

Our SVM vaulted verification scheme preserves privacy by having the client perform the SVM classification. This way, each classifier can be encrypted by the client's key; hidden from the server. To avoid placing trust on the client's authentication decisions, the server uses the classifiers to build the challenge, and the client must prove to the server that it actually has the feature vector being classified. Thus, only the client can decode or use the SVM classifiers, and only the client can test the feature vector. The server's role is to generate a token that the client can only derive from the proper SVM classifiers and the proper feature vector.

One way of generating such a token is to force the client to distinguish between "real" and "chaff" SVM classifiers, storing both kinds in the template. Each bit of the server's challenge-response token could contain one real and one chaff classifier. The client uses each classifier to classify the corresponding part of the feature vector and decides which classifier is real and which is chaff. The client's decision then becomes that bit of the response. This is the core of how vaulted verification works: storing multiple real and chaff SVM classifiers in the template and forcing the client to decide between the two at authentication time.

Because an attacker does not know the difference between the real and chaff SVMs, it cannot construct the response with better than random chance. Further, because each classifier is encrypted with a key that only the client knows, each template conceals the client's privacy from the server. This way, no identifiable information is stored on the client, and the data stored on the server cannot be decoded without the client's encryption key which never leaves the client's device.

Our protocol makes two important assumptions. First, we assume that a classifier can be chosen such that an honest subject can distinguish between the real and the chaff but an impostor cannot. Second, we assume that each feature vector can be split into meaningful parts such that each part is individually distinguishable from its corresponding chaff part. Both assumptions impose restrictions which could degrade performance. Other than that, this approach makes no real assumptions about features or classifiers – like fuzzy vaults, our approach is a very general concept.

The rest of this paper is devoted to a detailed description of the privacy-preserving SVM vaulted verification protocol at enrollment and match time. We then describe some security considerations and outline its implementation in a face authentication system, concluding with an evaluation of the performance and security of our system. Throughout the description of our protocol, we assume that all communication between the client and the server takes place under a trusted channel such as SSL with hard coded certificates.

5.1 Enrollment time

At enrollment, the client generates a feature vector, FV, that represents the user. This feature vector is then permuted (shuffled) according to a "permutation key," K_p, derived from the client's passphrase and/or secret key stored on the device to yield PFV. This permuted feature vector is then split into N smaller vectors, $s_1, s_2, \ldots, s_N$. Each "slice" of s has $\frac{1}{N}$ as many elements as PFV. Using s, N one-class SVM classifiers, $v_1, v_2, \ldots, v_N$, are trained. Because SVM classifiers generally require more than one sample in the training set, the same process can be repeated on multiple feature vectors of the subject, all permuted with K_p, to yield several s_i suitable for training classifier v_i. This allows a multi-view gallery to be used.

In addition to v, the client uses the same process to generate different but statistically indistinguishable "chaff" SVM classifiers $w_1, w_2, \ldots, w_N$ trained on corresponding "chaff" feature vectors. Ideally, these chaff feature vectors should be chosen such that the subject can tell which classifiers are elements of v and which classifiers are elements of w, but attackers will find them indistinguishable.

At this point, the client has N classifiers (v) trained on parts of real feature vectors from the client and N corresponding chaff classifiers (w) trained on the corresponding parts of chaff feature vectors.

The client then generates a polynomial, f, with $C < N$ random coefficients and a corresponding chaff polynomial g with C random coefficients. Even though f is never stored, it plays an integral part of the match process, used as both an authentication token and a means for error correction.

For each element v_i of v, the client creates the triple $(v_i, \;\; a, f(a))$ for some random number a. In a sense, this triple binds the serialized representation of classifier v_i to a random point on the polynomial. The client then creates the corresponding triple $(w_i,\ \ b, g(b))$, which binds the chaff classifier w_i to a point on the completely different, meaningless polynomial g. Each triple is then encrypted with an encryption function E with key K_e. Only the client knows K_e so only the client can recover the elements of each triple.

At this point, the template contains $\text{hash}(f)$ and N pairs of encrypted blocks:

$\text{hash}(f)$	
$E_1(v_1,\ \ a_1, f(a_1))$	$E_1(w_1,\ \ b_1, g(b_1))$
$E_2(v_2,\ \ a_2, f(a_2))$	$E_2(w_2,\ \ b_2, g(b_2))$
$\vdots$	
$E_N(v_N,\ \ a_N, f(a_N))$	$E_N(w_N,\ \ b_N, g(b_N))$

The client takes one last step before storing the template. If the template were stored as shown above, the server and attackers could know that v_i would always be the first element of each pair. To avoid this, the client defines a "canonical swapping," $CS = \{0,1\}^N$, derived from the coefficients of f. For each bit in CS (call it CS_i), if $CS_i = 1$ then the i^{th} pair is swapped in the template — the real becomes the second element and the chaff becomes the first. This way, the real and chaff are stored unpredictably.

Finally, the client encrypts the template with the server's public key, storing the final template on the client's device. This prevents an attacker from using the template if they steal it from the client and makes the server a less interesting target (see our security discussion in Section 5.3). To ensure that the client presents the correct template, the server stores only a hash of the encrypted final template.

After enrollment, what kind of information does each party have? From the client's perspective, this template contains a large block of data encrypted with the server's key. In order to authenticate, the client must send the template to the server who decrypts it before starting the rest of the protocol, discussed below. Because the client never has the "raw template" until after a successful authentication, an attacker cannot gain any useful information by stealing the template from the client.

From the server's perspective, once the client sends the template, the server can see that the template contains $\text{hash}(f)$ along with N pairs of encrypted triples; half of which encapsulate v and half of which encapsulate w. Because the server does not have K_e, it can never recover or alter the triples and thus it cannot obtain v. Further, assuming $\text{hash}(f)$ is one-way, the server cannot obtain the coefficients of f. These properties preserve the privacy of the client from the server and from any attackers that gain access to the classifier template.

5.2 Match time

5.2.1 Overall principle

During match time, the client first sends the encrypted template to the server. The server then generates an authentication token. In order to ensure that only the subject can recover this token, the server encodes this token as a unique transformation of the template and sends the transformed template back to the client, challenging the client to detect the transformation and extract the authentication token.

How can this transformation occur? Recall that the enroller encrypted each SVM with the client's key. The server clearly cannot change the *content* of the template, but it can change the template's *structure* by swapping the pairs of triples stored inside. This way, the server can consider each pair of triples as a single bit of a binary string of length N. The server can then encode this binary string by sending the template back with some pairs swapped. For each bit, if the server sends $(E_i(v_i, a_i, f(a_i)),\ \ E_i(w_i, b_i, g(b_i)))$, a binary 0 is assumed. If the server instead sends $(E_i(w_i, b_i, g(b_i)),\ \ E_i(v_i, a_i, f(a_i)))$, a binary 1 is assumed. Because the client can distinguish between v and w, the client knows which triple came first; thus, the true client can extract each bit of the string sent by the server. The client can then use this string to prove it actually had the biometric. Because an attacker cannot distinguish between v and w, they only have random chance of correctly guessing each bit.

5.2.2 Detailed protocol: Server's role

The server uses the steps outlined in Algorithm 1 to authenticate a client. To authenticate a client, the server receives the encrypted template from the client, compares its hash with the stored hash to ensure the client presented the correct one, and decodes it. The server then thinks of a random bitstring $B = \{0,1\}^N$ and a random $nonce$. B and $nonce$ are only valid for this session.

The client's ultimate goal is to return

$$\text{hash}(\text{hash}(f)\ ||\ nonce\ ||\ B)$$

where $||$ denotes concatenation. To do this, the client must extract the bits in B along with the coefficients of f through the matching process.

5.2.3 Detailed protocol: Client's role

At authentication time, the client captures a probe feature vector, FV', permutes it with K_p, and splits it into N parts, $p_1, p_2, \ldots, p_N$. Each element of p should be correctly classified by each corresponding classifier v stored on the server because at enrollment, v_i was trained with samples similar to p_i. The client decrypts m using session key K_s and saves the nonce. For each pair in m sent by the server, the client decrypts both triplets with K_e; call these triplets $(t_i,\ \ a_i, f'(a_i))$ and $(u_i,\ \ b_i, g'(b_i))$. Either $t_i = v_i$ and $u_i = w_i$ or $u_i = v_i$ and $t_i = w_i$. The client must decide

Algorithm 1 The server's authentication process.

```
$E(t) \leftarrow$ Receive encrypted template from client
Ensure $\text{Hash}(E(t))$ matches stored template hash
$hashf \leftarrow$ hash of $f$ from $t$
$pairs \leftarrow$ encrypted pairs of triples from $t$
$K_s \leftarrow$ a random Session Key
$nonce \leftarrow$ a random number
$B \leftarrow$ a random bitstring of length $N$
Initialize message to $m = nonce$
for $i = 0 \rightarrow N$ do
  $vblock, wblock \leftarrow pairs_i$
  if $B_i == 0$ then
    Append $(vblock, wblock)$ to $m$
  else
    Append $(wblock, vblock)$ to $m$
  end if
end for
Encrypt $m$ with $K_s$ and send both to client
$expected\_response \leftarrow \text{hash}(hashf \,||\, nonce \,||\, B)$
$actual\_response \leftarrow$ Receive from client
if $actual\_response == expected\_response$ then
  Authenticated
else
  Authentication_failure
end if
```

which is which by classifying each p_i with both t_i and u_i. For example, assume the classifier with the better classification score is v_i and the other one is the chaff classifier. The client could then retrieve the corresponding bit of B_i and assume that either $(a_i, f'(a_i))$ or $(b_i, g'(b_i))$ (depending on its choice) denotes a point on polynomial f.

f is over-determined because it has only C coefficients but the server sends N encoded points on the polynomial. This provides a sort of backwards error correction. If the client is unable to extract the required C points on f, it can perform a weighted search (such as that further described in Section 5.2.4) to find the correct polynomial.

Using this backwards error correction scheme, the client can obtain all the coefficients of f along with the bitstring response B'. Because of the canonical swapping, $B' \neq B$, but once the client has the coefficients of f, it can derive the CS used for enrollment. For each bit of CS, $B = CS \oplus B'$ The client can then send its response.

5.2.4 Searching the polynomial search space

The client must successfully extract all N bits of the bitstring to authenticate. To do this, he or she must correctly identify C bits and use the polynomial to recover the rest. This creates a problem: how does the client know which bits it guessed correctly?

In the detailed description of the protocol given above, we describe a naïve example in which the client decides each bit based on the assumption that v_i always yields a higher classification score on p_i than w_i. This assumption often does not hold. Thus, the client will have to search for the polynomial, using a backwards error correction approach wherein the SVM marginal distance provides a computational advantage over an adversary.

The weighted brute-force search works as follows: the server sends several pairs of triplets, call them $(t_i, \ \ a_i, f'(a_i))$ and $(u_i, \ \ b_i, g'(b_i))$. As usual, the problem is to find the coefficients of f by deciding which of t_i and u_i corresponds to v and which corresponds to w. To do this, the client sorts the pairs by "confidence," defined as the difference in classification scores, in descending order. Then, the client can take the C pairs with the highest confidence and discard the rest. The client can first try to construct f using the C most confident points. If this polynomial does not match the rest of the points in one element of each pair in the template, the client can try all possibilities of matches assuming one flipped bit. If no match is found, the client can try more possibilities with probabilities being driven by the SVM marginal distances. To verify any particular guess, the client can simply construct f' from the C pairs and find whether or not one element of the rest of the pairs lies on f'. If all pairs contain a point on f', the client knows that $f = f'$. This search strategy assumes that the client correctly guessed all but a small number of the least "confident" bits. To find the correct polynomial, the client must step through up to

$$\sum_{i=0}^{W} \binom{C}{i}$$

different possibilities, where W is the number of incorrect bits in the original guess. This scheme gives a large advantage to the honest subject who can correctly guess most of the bits and search the rest in order of likely errors. With a good chaff generation scheme (see Section 6.1.2), an attacker should get about half of the bits wrong due to random chance, and has no advantage in choosing what bits to flip while searching.

5.3 Security considerations

How does this SVM verification scheme satisfy the privacy requirements in 2? This model has many layers of security that work together to provide a private system.

First, the raw template is encrypted against the server key. The server only stores a hash of the template, making it an uninteresting target and deterring attackers. To decode the template, an attacker needs both the server's private key and access to the template stored on the client's device. Each block is individually encrypted with the client's private key, further preventing the attacker from recovering the raw classifiers. Even if an attacker decrypts v and w, the "canonical swapping" of each pair prevents the attacker from discerning between the real and the chaff without the coefficients of f, which would allow them to authenticate anyway. These properties prevent the server and attackers from acquiring the client's biometric data, preventing a compromised template attack.

Because the server cannot easily modify or decode the template in encrypted form, blended substitution and automated verification attacks are computationally infeasible. Also, because K_e and $\mathrm{hash}(f)$ are assumed to be unique to each enrollment, record multiplicity attacks are impossible and the client can revoke their biometric template by simply forgetting K_e.

During the authentication process, the only unique information exchanged between sessions is the server's nonce, the ordering of the encrypted template pairs, the hash of f's coefficients, and the hashed client response. None of these leak information about which pairs are which, so man-in-the-middle or eavesdropping attacks are fruitless, even assuming an attacker could subvert the SSL communication between client and server. To guess B with no a-priori knowledge and without K_e, the attacker would have one chance per authentication challenge to correctly guess all N bits along with all coefficients of f and would have no way of verifying their answer. Even if the attacker knew K_e, he would have to possess a biometric which eventually matched against v for at least C bits. If he obtains a similar biometric that correctly yields less than C bits, his search space would be greatly reduced depending on the degree of the biometric's similarity. If the attacker additionally wishes to compromise the client's privacy by obtaining a picture of the client's face, he would also have to obtain the permutation key K_p and then somehow derive a picture of the face from the reassembled feature vector. It is unknown if the SVM data could support image recovery.

6 Implementation in an SVM-Based Face Verification System

To test the merit of our SVM classification scheme, we implemented an SVM-based face verification system. The honest subject enrolls with face images and can then authenticate by presenting his/her face.

6.1 Implementation Details

6.1.1 GRAB Feature Vectorization

For every classification, we must convert images into feature vectors. We chose the GRAB descriptor as it has been shown to yield good accuracy on multi-class recognition problems in the FERET and LFW datasets [22]. GRAB feature vectorization works by transforming the image into a "processed" version of the image by using a method similar to linear binary patterns [22, 1]. The final feature vector is the concatenation of the histograms of each subwindow in the processed image. The goal here is not to show this is the optimal face-verification approach, but rather to validate the

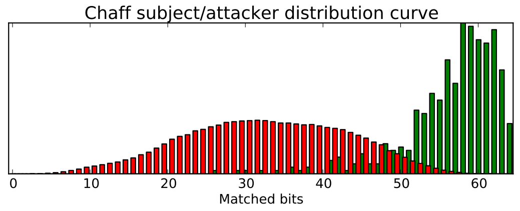

Figure 1. Histogram of how many bits impostors could identify (shown in red) versus how many bits the subject could verify (shown in green). Ideally, attackers guess about half of the bits (random chance) and the subject correctly guesses most of them.

vaulted verification design and understand the impact of the vaulted-verification challenge-response model on the original SVM's accuracy.

During matching and enrollment, the feature vectors are permuted according to K_p and split into N parts for SVM training. In this scheme, permuting the feature vector has the effect of "spreading" the important discriminatory information about areas near the eyes, mouth, etc. across multiple SVM classifiers inside the template, thus improving overall accuracy. For these experiments, we use one-class RBF SVMs as implemented by libsvm [5].

6.1.2 Chaff generation

A good chaff generation scheme is an important part of our protocol. If w is "too close" to v, the subject may have a hard time discerning them. On the other hand, if w is "too far" from v, an attacker might be able to gain access since many faces might be closer to v than w.

In our face verification system, we trained w on *random parts of feature vectors of other people*. This required a change to the experimental protocol: At the beginning of our test, we took 64 subjects from the dataset and removed them from the rest of the experiment. Each image from these subjects was processed using the same permutation K_p and used to train $64 \times N$ SVM classifiers to yield several "canonical negative examples" $E_1 \ldots E_{64}$. Each element of w was the corresponding slice of one example from E, chosen randomly. Practically, using a limited set of "negative examples" makes the chaff easier for an attacker to guess if they gain access to the pool of examples used, so a much larger pool should be used in practice.

To evaluate these ways of generating chaff, we ran experiments using the evaluation protocol in Section 6.2.1. No searching was done and no polynomial was used to correct incorrect bits. The purpose of this evaluation was to identify the distribution of scores for this chaff generation scheme. Figure 1 describes the subject's advantage over the attackers in discerning each real SVM from the chaff in terms of the distribution of how many bits the subject correctly guessed versus how many bits the attacker correctly found. Note this presumes ground-truth knowledge but highlights the advantage for the true subject.

6.2 Evaluation

6.2.1 Experimental Protocol

Because our experiments depend on machine learning classifiers that require three or four images to train each subject, we could not use the standard FERET protocol because each subject may have only a single gallery sample. Instead, we chose the FERET240 set, a subset of FERET [19] that contains the subjects that have at least four images [22]. While FERET240 has been used for identification testing, we adapt it for verification testing in the following manner. We choose 64 subjects to be the "chaff generators". Three pictures of a subject were chosen for their gallery and their remaining images were positive-labeled probes. All images for the other 175 subjects were impostor images.

To establish a baseline, we ran the experiment as described above with a single SVM classifier classifying a GRAB feature vector. The original GRAB algorithm combined the 64 regions' features, but here we classify each separately. To improve privacy we not only use the 64 regions of the original grab, but we also include a random permutation of the features before training the 64 classifiers.

6.2.2 Results

For the first baseline – a single one-class classifier trained on raw GRAB feature vectors – we achieved 57.2% TAR at 0.0107% FAR. The split-region version with $N = 64$ classifiers trained on parts of the feature vector, is our primary baseline. SVMs don't lend themselves well to varying security/convenience type analysis, but with a multiple classifier model we can easily vary the number of matching bits. At 56 bits matching, the baseline performance was .08%FAR and 69.5%TAR. For higher convenience settings, the baseline at 48 bits provided a FAR of 6.89% at a TAR of 90.15%. Note the split region baseline is considerably weaker than the baseline set by a "vanilla" face verification system, so there is an inherent loss in simply combining counting "bits" compared to the power of an SVM combining the regions.

Note that using only 3 images for training is minimal for a 1-class SVM approach. As a personal verification system, we expect many many more images, which would significantly improve TAR performance at a fixed FAR. However, our goal here is understanding the valuated verification performance impact and using existing datasets will improve scientific reproducibility. Future work will test with more appropriate datasets, including liveness detection.

The rest of our tests used the protocol described in Section 6.2.1 with the "random negative example" chaff scheme and the polynomial search described above, ending the backward error correction search after 1024 steps. We tested vaulted verification with various numbers of required matched bits (C), and varied the number of error correction steps as a means of varying the FAR. To measure

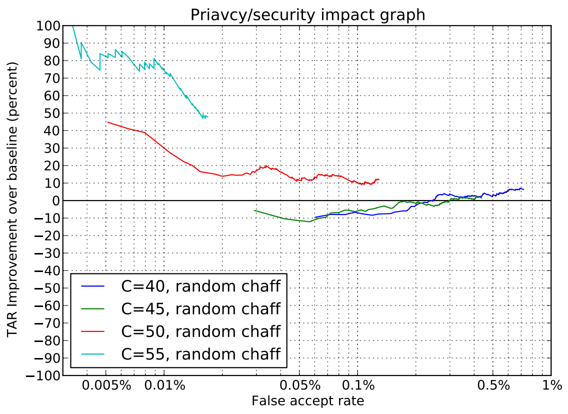

Figure 2. Privacy/security impact graph, showing % improvement in TAR over split baseline on Y-axis given a certain FAR. Similar to an ROC, threshold was the number of polynomial search steps required to find f. N=64, C=35

our performance relative to the split baseline, we create a "Privacy/security impact graph" which shows the percent improvement/loss over the baseline TAR with respect to a given FAR, as shown in 2. This graph shows the relative gains and losses that our privacy protection has at various FAR, plotting $100 * \frac{V-B}{B}$ where V if the TAR of Vaulted Verification at the given FAR, and B is the TAR of the baseline algorithm at that FAR. We can see that at larger FAR rates, vaulted verification shows slightly decreased performance, and at lower FAR (higher security), vaulted verification actually improved performance.

Our most secure vaulted verification test required 55 bits to match to find f. While this test significantly improved performance over the split baseline, the TAR is likely too low to be feasible, though with more training images it may become viable. Our least secure vaulted verification test only required 35 bits, but under-performed the baseline achieving an FAR of 4.20% at a TAR of 77.5%. This fell well below the split baseline, underscoring the importance of choosing sensible parameters when implementing Vaulted Verification.

6.2.3 Security considerations

Keep in mind that these impostor trials only tested the biometric's ability to resist a stolen template attack. We essentially presume the attacker has both the server and the client's encryption keys/passwords, testing only the biometric performance of our algorithm separately from its security. For these tests, we assumed the attacker somehow acquired K_e, and K_p, which requires physical access to the client's device and the client's passphrase. If the attacker could get this far, they could likely find an image of the client's face, thus compromising the biometric. If the attacker did not know K_e and K_p, there would be no false accepts because guessing f and B is statistically infeasible; see Section 5.3.

7 Conclusion and Future Work

This paper discussed the need to create privacy-preserving remote SVM verification systems. To achieve this, we devised new biometric-based challenge-response protocol that allows the client to perform the matching while proving its authenticity to the server. We implemented our SVM Vaulted Verification protocol in an SVM-based face recognition system. We presented preliminary results that demonstrate this protocol's feasibility for face verification and we outlined several possible ideas for improvement.

In the real world, more than 64 bits of security are desired. Thus, one objective of future experiments will be to evaluate our privacy-preserving SVM verification protocol at higher N and multiple C.

Future work will also involve improving accuracy and applying this research beyond face matching, extending it to other domains such as voice, keystroke, and iris verification. It will also address the very important aspect of liveness detection, ensuring the system cannot verify against a static image.

8 Acknowledgments

The research reported in this document has been funded partially by NSF PFI Award #065025, NSF grants CNS-0958576 and CNS-0851783.

References

[1] T. Ahonen, A. Hadid, and M. Pietikäinen. Face description with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:2037–2041, 2006.

[2] A. Beng Jin Teoh and C. T. Yuang. Cancelable biometrics realization with multispace random projections. *IEEE Trans. Syst. Man Cybern. B, Cybern.*, 37(5):1096 –1106, oct. 2007.

[3] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp 245–250, 2001.

[4] T. Boult. Robust distance measures for face-recognition supporting revocable biometric tokens. *IEEE Int. Conf. Automatic Face and Gesture Recognition*, pp 560–566, 2006.

[5] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2:1–27, 2011.

[6] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *ICDM*, pp 589–592, 2005.

[7] M. Crompton. Biometrics and privacy the end of the world as we know it or the white knight of privacy? *Australian J. Forensic Sciences*, 36(2):49–58, 2004.

[8] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, volume 5672 of *LNCS*, pp 235–253. Springer, 2009.

[9] Y. C. Feng, P. C. Yuen, and A. K. Jain. A hybrid approach for face template protection. In *SPIE*, volume 6944, 2008.

[10] S. Guo and X. Wu. Deriving private information from arbitrarily projected data. In *Advances in Knowledge Discovery and Data Mining*, volume 4426 of *LNCS*, pp 84–95. Springer, 2007.

[11] S. Jassim, H. Al-Assam, and H. Sellahewa. Improving performance and security of biometrics using efficient and stable random projection techniques. In *Proc. 6th ISPA*, pp 556 –561, 2009.

[12] A. Juels and M. Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 38:237–257, 2006.

[13] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *ACM Conf. Computer and communications security*, pp 28–36, 1999.

[14] Y. Kim and K.-A. Toh. A method to enhance face biometric security. In *IEEE 1st Int. Conf. Biometrics: Theory, Applications, and Systems*, 2007.

[15] Y. Li, S. Gong, J. Sherrah, and H. Liddell. Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, 22(5):413–427, 2004.

[16] K.-P. Lin and M.-S. Chen. Releasing the svm classifier with privacy-preservation. In *IEEE Int. Conf. Data Mining*, pp 899–904, 2008.

[17] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. Knowl. Data Eng.*, 18:92–106, 2006.

[18] P. Mihailescu, A. Munk, and B. Tams. The fuzzy vault for fingerprints is vulnerable to brute force attack. In A. Brmme, C. Busch, and D. Hhnlein, editors, *BIOSIG*, volume 155 of *LNI*, pp 43–54. GI, 2009.

[19] P. Phillips, H. Moon, S. Rizvi, and P. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(10):1090–1104, 2000.

[20] N. Pinto, J. J. DiCarlo, and D. D. Cox. How far can you get on a modern face recognition test set using only simple features? *IEEE Conf. CVPR*, 2009.

[21] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle. Generating cancelable fingerprint templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:561–572, 2007.

[22] A. Sapkota, B. Parks, W. Scheirer, and T. Boult. Face-grab: Face recognition with general region assigned to binary operator. In *IEEE Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp 82 –89, 2010.

[23] W. Scheirer and T. Boult. Cracking fuzzy vaults and biometric encryption. In *Biometrics Symposium, 2007*, 2007.

[24] H. Sohn, Y. M. Ro, and K. Plataniotis. Biometric authentication using augmented face and random projection. In *IEEE Int. Conf. Biometrics: Theory, Applications, and Systems*, 2009.

[25] H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proc. ACM Symp. Applied Computing*, pp 603–610, 2006.

[26] H. Yu, J. Vaidya, and X. Jiang. Privacy-preserving svm classification on vertically partitioned data. In *Advances in Knowledge Discovery and Data Mining*, volume 3918 of *LNCS*, pp 647–656. Springer, 2006.