Vaulted Verification: A Scheme for Revocable Face Recognition

Michael Wilber
University of Colorado, Colorado Springs
mwilber@uccs.edu

***Abstract*—As biometric authentication systems become common in everyday use, researchers are beginning to wonder about the ethics of biometric recognition. In particular, this paper outlines the need for a face recognition system that does not compromise the privacy of the subjects being recognized. We present a brief overview of current face verification systems and discuss one such implementation. We also outline several obstacles that must be overcome to protect SVM-based face classifiers. To overcome these obstacles, we present a novel protocol we call "Vaulted Verification" that allows a server to authenticate a client's biometric in a privacy preserving way. Finally, we conclude with a small evaluation of performance, discussion of some security implications, and ideas for future work.**

I. Introduction

FACE recognition systems are ubiquitous. The real-world usefulness of a system that recognizes faces combined with increasing computational power has led to an explosion of research over the past few years. As a result, many different face recognition systems enjoy widespread use in several professional domains such as academia, security, data mining, and information retrieval.

Given their ubiquity, it is important that face recognition systems can both guarantee security and operate without sacrificing individual privacy. This project seeks to build an accurate, secure, and private face verification system. By "accurate", we mean that this system should correctly recognize faces with no false positives. By "secure" and "private", we mean an individual should be able to revoke their identification at any time and an attacker should not be able to reconstruct information about the face if they somehow acquire a copy of the template database. Finally, this system should operate well under the speed and memory constraints imposed by modern hardware.

Unfortunately, there are few existing verification packages available that preserve privacy and security. Corporations are beginning to grapple with the dilemma of ethical face recognition. With increased widespread use comes increased responsibility and increased pressure to preserve customers' privacy. Recent events such as Facebook's controversial widespread deployment of automatic face recognition technology [1] are beginning to raise questions about privacy in customers' minds.

II. Existing face recognition schemes

According to [2], there are three closely related problems in the domain of automated face recognition:

- **Pair matching**. In this scheme, an algorithm decides whether or not two separate photos are of the same individual.
- **Recognition**, also known as identification, closed-set, or "multi-class" recognition. A gallery is constructed containing photos of several different people. The algorithm takes a new photo of a subject and decides who it is, selecting one out of the set of gallery subjects.
- **Verification**, also known as authentication, open-set recognition, or "one-class" recognition, is where the gallery consists of several images of a single subject. The algorithm is given either an image of the subject or an image of an impostor from an unknown universe of impostors. The algorithm then decides whether the probe image is of the subject in the gallery or is an impostor.

We concern ourselves with the last of these problems as it is the issue addressed by most common biometrics systems. Our face verification system is analogous to a fingerprint scanner in the sense that it can be used to allow or deny an individual's access to a resource.

How can one build a face verifier that does not compromise privacy? To answer this question, we will first present an overview of current face verifiers. Then, we will describe the privacy issues in Section III and present the preliminary results of a novel protocol designed to overcome these issues.

Consider the following scenario: User Bob wishes to log in to his bank account using his face. When Bob visits the bank to enroll, his bank takes several pictures of his face. They normalize these pictures and convert them into feature vectors which are stored in the bank's gallery database. To authenticate a potential account holder's photograph (probe), the stored feature vectors are used to train a one-class SVM classifier which then either accepts or rejects the probe.

A. Experimental protocol: Dataset composition

Several commonly used public-accessible datasets exist for the purpose of studying facial recognition problems. In particular, FERET [3] and LFW [2] are among the most recognized, but they answer slightly different questions as FERET is intended for closed-set recognition [3] and LFW is primarily concerned with image pairing [2].

Because our experiments depend on machine learning classifiers that require three or four images to train each subject, we chose the FERET240 set, a subset of FERET that contains the subjects that have at least four images, as described in [4].

FERET240 is intended to test recognition problems rather than verification problems (see Section II), so the following changes were made to the test protocol: First, the test is repeated for each of the 240 subjects. The total scores – true and false positives and rejects – are summed and considered as scores of the overall test. g pictures of the subject (usually three or four depending on test) are considered to be the gallery, and the rest of the subject's pictures are considered the positive-labeled probes. The impostors are all images of the other 239 subjects in the set.

B. Generating the feature vector: GRAB

For every classification, we must convert images into feature vectors. For this project, we will use the GRAB descriptor [4] as it has been shown to yield good accuracy on multi-class recognition problems in the FERET and LFW datasets. GRAB has also been used on open-set recognition problems similar to the ones we face [5].

GRAB is a modification of linear binary patterns (LBP) [4]. LBP works by splitting the image up into 64 sub-windows [6]. A "feature histogram" is computed for each sub-window, where each feature is found by thresholding each pixel's brightness against its eight neighbors, yielding an 8-bit number. This means each pixel may possess one of 256 distinct feature "tags". Each sub-window's feature histogram is then concatenated to form the final feature vector.

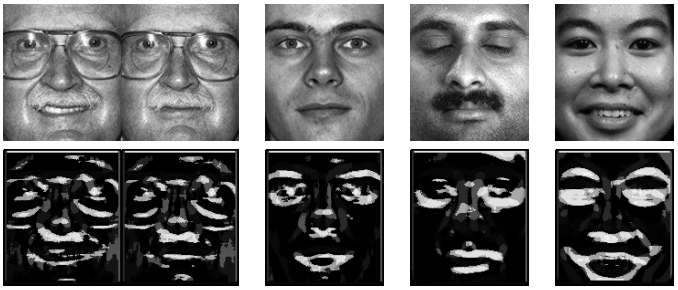

Fig. 1. A demonstration of GRAB neighbor preprocessing before windowed histogram application. The two images on the left are of the same subject and demonstrate intra-class variation; the three images on the right are of different people and show inter-class variation.

GRAB works in much the same way but with some slight differences. First, instead of sampling each pixel's neighbors, GRAB samples different-sized neighborhoods of pixels around the target pixel. This makes GRAB less resistant to noise and resolution changes than LBP [4]. Second, GRAB demands that each pixel's brightness must be at least a given threshold brighter or darker than its neighborhood, whereas LBP only compares the two pixels' values by a simple "greater-than" or "less-than" comparison. This ensures GRAB only finds features important enough to yield high contrast changes.

C. Classification

Unfortunately, GRAB feature vectors may have nonlinear intra-class variations. This means it is often impossible to distinguish between two vectors of the same subject versus two vectors across different subjects with a "nearest neighbor" classifier. To work around this, machine learning techniques such as support vector machines (SVM) are often used because they can better distinguish these relationships [4], [7].

A binary SVM is trained against positive and negative feature vectors. The output is a support vector machine that, when given an unknown feature vector, can distinguish whether it is a member of the positive class or the negative class. Because our problem involves verification rather than recognition, we wish to use an SVM that can distinguish between the intended subject and everyone else. Unfortunately, a binary classifier is unfeasible because we cannot possibly know all of the negative examples – this would require taking pictures of every possible impostor. Thus, we use a one-class SVM, which can distinguish between members and nonmembers of the gallery set.

D. Preliminary evaluation results

We built the above system to establish a good baseline of the kind of results we can expect. This baseline will show us errors in our existing implementation and will reveal how privacy-preserving mechanisms degrade verification scores.

	TAR	FAR
One-class SVM, linear kernel	71.09%	39.38%
Gaussian kernel (γ = 0.5)	78.80%	4.51%
Gaussian (γ = 0.75)	72.38%	3.26%
Gaussian (γ = 1)	65.73%	2.85%
Gaussian (γ = 2)	57.39%	2.41%
Gaussian (γ = 5)	50.96%	2.10%
Open-set SVM (γ=1, η=best rejected score)	80.09%	1.04%
η=$\frac{9}{10}$ between rejected and accepted	64.6%	0.017%
η=Optimize recall when precision=.75	81.80%	0.89%

TABLE I
PRELIMINARY RESULTS OF A ONE-CLASS SVM VERIFICATION SCHEME. TAR, FAR = TRUE ACCEPT RATE AND FALSE ACCEPT RATE

In Table II, a "true positive" is defined as correctly accepting an honest subject and a "false positive" is defined as accidentally accepting an impostor. True positive and false positive rates are obtained by taking the corresponding statistic and dividing it by the respective number of classifications that should have been accepted and rejected – for this experiment, 467 classifications should have matched and 251,906 should have been rejected.

At a fundamental level, it is hard to pick the proper parameters for a one-class SVM to achieve a desired level of generality. Should the SVM match against all images? All humans? All humans of a certain ethnicity? Images of a certain subject? Only images of a certain subject with a certain pose? Depending on the problem, all of these may be desired classifiers. Very recent work with "open-set support vector machines" is beginning to emerge as a possible solution. Open-set SVMs build upon one-class SVMs by using "margin morphology" techniques to achieve a desired level of generality by shrinking or expanding the 1-class SVM's decision function to minimize error [5]. This essentially works by training a one-class SVM as usual and then providing canonical negative examples. These do not impact the support vectors or kernel parameters (as they would in a binary SVM);

instead, these canonical negatives provide a reference for the desired generality of the classifier. Three experiments with an open-set SVM classifier are included in Table II.

III. Privacy concerns of SVM-based recognizers

Implementing a face verifier as per the above system may satisfy the "accuracy" requirement in that the system will ideally accept only the subject and reject impostors (please humor us), but this system does not address the "security" or "privacy" requirements. In other words, simply building an SVM-based classifier is not enough to ensure the privacy of the subject being classified. In our case, we want to protect the final classifier itself because if an attacker could obtain the SVM representation, he could conceivably consider local maxima of the kernel function to produce a feature vector that would be accepted by the SVM. Once the attacker has suitable feature vectors, he could conceivably produce an image that has the same GRAB features. Even if this image might not look anything like the original subject, it would still be falsely accepted because it would evaluate to a feature vector that the kernel classified in the same way as the subject.

The literature discusses several ways of "protecting SVM privacy", but none of them are directly suitable to this problem domain.

A. Previous approaches to SVM privacy protection

[8], [9] describe two secure ways of training an SVM where multiple parties have different shards of the training set. Unfortunately, both methods are secure only when there are more than three parties. Also, though these methods protect the training set, they do not address protecting the SVM itself after training; it is assumed to be securely stored by a third party – this is an assumption we cannot make.

[10] describes a way of post-processing a trained SVM to protect privacy of the support vectors, 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 still leaks 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 only slightly altered [10] or completely unprotected [8], [9].

B. Fuzzy Vaults: Another approach to biometric privacy

Other previous work involving privacy-preserving biometric matching includes the use of "fuzzy vaults", where a secret can be 'locked' in the coefficients of a polynomial and can only be 'unlocked' if the subject provides a certain number of biometric features. In theory, privacy is preserved by including several 'chaff' features. The security of the scheme rests on an attacker's inability to guess the real features among the chaff.

Unfortunately, the literature contains both practical security problems of fuzzy vaults [11] (outlined in Section III-C) and information-theoretic weaknesses in certain implementations of fuzzy-vaults [12], [13]. Our protocol includes many ideas from fuzzy vaults but used in unconventional ways. As such, we try to sidestep many of these issues.

Fuzzy vaults [14], an improvement of "fuzzy commitment" [15], are cryptographic schemes that allow a user to conceal a secret S using a set of elements A as a key. S can be decoded by obtaining a sufficient number of elements of A. These vaults are "fuzzy" because not all elements of A are required to release S, the elements of A used to lock and unlock S can differ by a small amount, and the elements of A can come in any order. This is useful for biometric systems where noise, reordering, and missing information are obstacles that must be amended.

In brief, fuzzy vaults work by embedding the secret S inside the coefficients of some polynomial F. To lock this secret with set $A = \{A_1, A_2, \ldots, A_k\}$, a list of pairs is built: $(A_1, F(A_1)); (A_1, F(A_2)), \ldots, (A_k, F(A_k))$. Additionally, chaff points are added to this list by concatenating pairs of random numbers $(R_1, R_2), (R_3, R_4), \ldots$ to the list such that $R \cap A = \emptyset$ and $R \cap \{F(A_i) | A_i \in A\} = \emptyset$. The list of pairs is then permuted to remove ordering information.

Because an honest user has a sufficient subset of A, they can easily tell which pairs are chaff and which pairs correspond to $(A_i, F(A_i))$. From this, they can determine the coefficients of F and decode the secret S embedded therein.

C. Shortcomings of Fuzzy Vaults

Unfortunately, using fuzzy vaults alone may not provide adequate security for all applications. [12] explores the insecurity of certain configurations of fuzzy vaults by presenting the feasibility of better-than-brute-force attacks. Further, [11] lists three potential attacks with potentially serious consequences:

- An **attack via record multiplicity** allows an attacker to correlate the subject's records across multiple vaults by comparing the pairs of $(A_i, F(A_i))$ across each vault. Common (or similar) pairs are more likely to be elements of A and pairs that are dissimilar are likely to be chaff. At best, the attacker's search space is greatly reduced. At worst, the attacker can piece together the elements of A and by solving for F, the attacker can trivially decode both secrets.
- A **surreptitious key inversion** attack allows the attacker to recover the elements of A if he knows S. By building a polynomial from the coefficients of S, he can trivially determine which points lie on F and which are chaff. From there, he can recover A. If the elements of A are raw biometrics, the attacker has compromised the subject's privacy.
- An **insidious substitution attack** allows an attacker to construct a fuzzy vault that silently authenticates both him and the subject. This can be done without alerting the subject that anything is wrong. To do this, the attacker edits the stored fuzzy vault and replaces the chaff points $(R_1, R_2), (R_3, R_4), \ldots$ with $(B_1, F(B_1)), (B_2, F(B_2)), \ldots$ for his own set B. This constructs a fuzzy vault where $(B_i, F(B_i))$ look like chaff from the subject's point of view and $(A_i, F(A_i))$

look like chaff to the attacker's point of view. Thus, the vault authenticates both subject and attacker without any warning to the subject. Now the attacker can log in to the subject's account without the subject's or the bank's knowledge.

IV. Evaluation

To find out how well our Vaulted Verification protocol performs, we implemented a preliminary version over the course of the summer.

	TAR	FAR
N=64, Fuzzy search, chaff from random person, required bits = 20	70.491%	6.549%
N=64, Required bits = 35	38.235%	0.508%
N=64, Required bits = 45	36.364%	0.052%
N=64, Required bits = 50	23.529%	0.000%

TABLE II
PRELIMINARY RESULTS OF THE VAULTED VERIFICATION PROTOCOL AS APPLIED TO FACE RECOGNITION. IN THIS TABLE, TAR MEANS TRUE ACCEPT RATE AND FAR MEANS FALSE ACCEPT RATE.

Table II presents preliminary results of several tests at varying levels of sensitivity. Unfortunately, each test is very computationally expensive and thanks to a power outage partway through, these results are representative but incomplete.

Each test follows the experimental protocol established in Section II-A. For these tests, "required bits" depicts how many bits of the challenge are required to authenticate. The total template contained 64 bits. This allowed us to vary the sensitivity.

In the table presented, the highest true accept ratio required 20 bits. This yields only 1,048,576 different possible keys and is feasible for a brute-force search. In the real world, we would require many more than 64 bits for an attacker to guess. The next stage of our work involves trying larger-scale experiments. Other improvements can also be made. For example, several feature vectors of the subject can be used at match time to improve confidence.

These impostor trials assume the attacker successfully subverted SSL encryption along with all keys kept by a client. These tests only test how well the biometric itself protects the template and in a practical sense, a successful authentication requires physical access to the client's device. If the attacker could get this far, they could likely obtain an image of the face and authenticate easily anyway. That said, without working face recognition, we have merely re-implemented two-factor authentication. There are clearly many opportunities for future improvement here.

V. Goals, Status, and Roadmap

This phase of the project is nearing completion. Over the course of the summer, we implemented the vaulted verification protocol on top of the MFFR experiment framework. This allows for easy experimentation and sweeping customizations to the experimental protocol. `mffr` is a modular facial recognition pipeline descended from the "V1" codebase [17], [18]. Our first task was to reproduce the GRAB experiment [4] with the mffr pipeline as a quick sanity check. To do this, we reimplemented GRAB in Python. This created a unified pipeline for the future work of other lab mates. Our implementation of the GRAB feature vectorization was pixel-perfect with `grab-c`, the reference implementation. Though we reimplemented the experiment as described, we achieved inferior (but comparable) results and are working with the author to improve recognition rates. This may yield improvements for other areas of our pipeline as well.

Because mffr was only suited to recognition problems, as the next step, we adapted mffr to handle face verification problems as well. After all, the vaulted verification protocol is a verification problem at heart, not a multiclass recognition problem.

Once this was completed, we ran preliminary experiments of an ordinary GRAB-based face verifier, the results of which are presented in Table II. This provided a baseline of how well a non-privacy-protecting scheme would work.

Date due	Task description	Status
—	Implement GRAB feature vector descriptor in mffr pipeline	**Done**
—	Reproduce GRAB recognition experiment [4] with mffr pipeline (as a sanity check)	**Done** (imperfect)
—	Repurpose mffr pipeline for verification challenges	**Done**
—	Run preliminary experiments, gather baseline with one-class SVM	**Done**
—	Optimize/improve baseline	**Done**
—	Implement Open-set SVM	**Done**
—	Phase 1: Vaulted Verification protocol, first iteration	**Done** (imperfect)
Aug 5	Final presentation and paper	**Done**
Aug 15	Phase 2: Vaulted Verification protocol, refine and harden	To do
Sept 15	Final ICB2012 submission deadline	On track

TABLE III
SCHEDULE AND TARGET DATES

We then implemented the first draft of the vaulted verification protocol, collecting preliminary results as seen in Table II. This was intended to find various ways of defining chaff and their impact on recognition scores.

The final product of this work will be a paper submitted to ICB 2012, the 5th International Conference on Biometrics. The submission deadline is September 15, and we hope to have all work completed by then. This paper will describe the details and implementation of our face verification system. Time permitting, we will also describe possible approaches to fingerprint verification.

Long-term goals of this work include commercialization of a privacy-enhanced face recognition system, its implementation on mobile devices, and searches for further ideas of improving the privacy protection scheme while raising recognition rates.

VI. Conclusion

This paper demonstrated the need for a face recognition system that earns users' trust by allowing faces to be recognized without knowing what those faces look like. To do this, we started out by building a normal, ordinary face verification

system by using GRAB features and one-class SVMs. We then implemented the "vaulted verification" protocol as a way of protecting subjects' privacy. We presented preliminary results that demonstrate this protocol's feasibility for face verification and we outlined several possible ideas for improvement.

ACKNOWLEDGMENTS

The research reported in this document has been funded partially by NSF grants CNS-0958576 and CNS-0851783.

REFERENCES

[1] "Facebook 'face recognition' feature draws privacy scrutiny," ser. Bloomberg News. New York Times, 2011.

[2] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[3] P. Phillips, H. Moon, S. Rizvi, and P. Rauss, "The feret evaluation methodology for face-recognition algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22(10), pp. 1090–1104, 2000.

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

[5] Anonymous, "Margin morphology and the open set svm," Currently under review for ICCV, Tech. Rep., 2011.

[6] T. Ahonen, A. Hadid, and M. Pietik?inen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 2037–2041, 2006.

[7] D. Fradkin and I. Muchnik, "Support vector machines for classification," Series in Discrete Mathematics and Theoretical Computer Science, Tech. Rep., 2006.

[8] H. Yu, X. Jiang, and J. Vaidya, "Privacy-preserving svm using nonlinear kernels on horizontally partitioned data," in *Proceedings of the 2006 ACM symposium on Applied computing*, ser. SAC '06. New York, NY, USA: ACM, 2006, pp. 603–610.

[9] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving svm classification on vertically partitioned data," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, W.-K. Ng, M. Kitsuregawa, J. Li, and K. Chang, Eds. Springer Berlin / Heidelberg, 2006, vol. 3918, pp. 647–656, 10.1007/11731139_74.

[10] K.-P. Lin and M.-S. Chen, "Releasing the svm classifier with privacy-preservation," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 899–904.

[11] W. Scheirer and T. Boult, "Cracking fuzzy vaults and biometric encryption," in *Biometrics Symposium, 2007*, sept. 2007, pp. 1 –6.

[12] P. Mihailescu, A. Munk, and B. Tams, "The fuzzy vault for fingerprints is vulnerable to brute force attack." in *BIOSIG*, ser. LNI, A. Brmme, C. Busch, and D. Hhnlein, Eds., vol. 155. GI, 2009, pp. 43–54.

[13] E.-C. Chang, R. Shen, and F. W. Teo, "Finding the original point set hidden among chaff," in *ASIACCS*, 2006, pp. 182–188.

[14] A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, pp. 237–257, 2006, 10.1007/s10623-005-6343-z.

[15] A. Juels and M. Wattenberg, "A fuzzy commitment scheme." ACM Press, 1999, pp. 28–36.

[16] V. Guruswami and M. Sudan, "Improved decoding of reed-solomon and algebraic-geometric codes," in *Foundations of Computer Science, 1998. Proceedings.39th Annual Symposium on*, nov 1998, pp. 28 –37.

[17] B. Parks, "Mffr (multi-feature face recognition) user manual," University of Colorado at Colorado Springs, Tech. Rep., 2011.

[18] 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 Computer Vision and Pattern Recognition (CVPR)*, 2009.