PRIVV: Private Remote Iris-authentication with Vaulted Verification

Michael J. Wilber, Walter J. Scheirer, Terrance E. Boult
University of Colorado at Colorado Springs and Securics, Inc.
Colorado Springs, CO, 80918, USA
{Last name}@vast.uccs.edu

Abstract

Iris biometrics are one of the strongest modalities in terms of performance, and as such, are used in several high-security scenarios. However, remote/network authentication creates its own class of problems that have not previously been considered. In this work, we adapt Vaulted Verification, a novel remote matching protocol, to work with iris biometrics in a network authentication setting. We give an overview of how Vaulted Verification works and show how to adapt it to the unique challenges of iris biometrics. We then investigate the algorithm's security and evaluate its performance, showing that Vaulted Verification is among the top-performing template protection algorithms for the CASIAv1 dataset. We conclude with a summary and ideas for future work.

1. Introduction

Iris matching systems are seeing increasingly common use; as such, it is important that they operate efficiently and accurately. However, they must also take care to address privacy issues, especially in a network-aware setting. This paper presents a new approach to building an accurate, private, and secure iris verification system, capable of remotely matching a subject's iris without compromising their privacy.

Consider Chris (C), the client, who wishes to biometrically verify his identity using, say, his mobile phone and/or passport which contains an iris template. Chris has concerns about his privacy and how any server, such as his bank, uses/stores his biometric. For instance, Chris, having read [5], wishes the matching to be done on his phone, but he also wants to ensure that the design prevents attackers from gaining access to his biometrics—he only has two irises, and he cannot change them. Chris also wants to be able to cancel his template simply by forgetting his password, and his canceled template should be worthless to anyone who finds it.

The server to which Chris is authenticating, say B (e.g. for bank) has security concerns. Ideally, their authentication system should properly resist man-in-the-middle and replay attacks, preventing attackers from reusing stale authentication data. They also wish to ensure that no one else can use Chris' identity, whether intentionally or unintentionally; accepting impostors is also a significant security risk. For B, it must be possible for all of these requirements to be met without a significant drop in accuracy of the biometric verification. As B does not control the matching process, it needs strong assurance that a high-quality match happened in Chris' device, even if Chris wishes to share his identity with a friend. Traditional iris biometrics cannot be used directly because these require Chris to send his iris to the server or for the server to blindly trust his phone's yes/no decision or match score.

Our main novel contribution is creating a client/server protocol for iris-based authentication that addresses these issues while increasing performance. In fact, this protocol provides an increase in accuracy over the baseline system. To do this, we adapt "Vaulted Verification," introduced in [21], to work with iris codes. We then investigate this protocol's security from an attacker's point of view and evaluate its performance on the CASIAv1 iris dataset.

2. State-of-the-Art Private Iris Verification

Iris verification is a growing field, and many advances have already been made in the realm of privacy protection for Iris templates. Some previous techniques are based on random projection, applying a randomly-generated orthogonal matrix to an iris feature vector. This is a non-invertible transformation that discards information, and as such, sacrifices performance. [13] is one such example that randomly projects segments of the iris data. More recently, [14] improves on earlier work, reporting TAR=98.13 at FAR=0.001 on the ICE2005 dataset. Unfortunately, techniques based on random projections may be compromised by ICA-based attacks, and as such, may have unsolved security issues [3, 9]. Other transform schemes such as [15, 24] reduce iris data in such a way that they significantly reduce performance compared to the baseline algorithm.

Because iris codes are generally represented as bit vectors, there are many groups that have looked at mappings between iris codes and cryptographic keys. Some privacy-enhanced iris verification systems bind pre-existing cryptographic keys with iris codes ("key-binding biometric cryptosystems"), while others derive cryptographic keys from the iris code itself ("key-generating biometric cryptosystems"). Both systems typically use error correction to account for the inherent variability of the iris, but to date none have acceptable accuracy.

An example of a key-generating system, [4] uses a "fuzzy extractor" scheme deriving a key from an iris code and helper data. The performance is weak—TAR=88.9% when FAR=1.35% on a small subset of CASIA.

There is much research in the area of key-binding iris cryptosystems. Fuzzy commitment schemes such as [1, 22] bind the iris with a cryptographic key expressed as an error correcting code. However, these techniques have security problems; [16, 23] presents attacks against fuzzy commitment schemes. According to [23], one reason that makes fuzzy commitment weak for irises is because iris codes generated from Gabor features can be seen as Markov models—the bits in an iris code are not uniformly independently distributed, allowing an attacker to exploit the error correction to their advantage. Some schemes shuffle the iris codes to remove the local dependency of bits [7]. This helps better separate the inter-class and intra-class match score distributions and also enables revocability, if the key for shuffling is an un-stored user secret. [23] reports that permuting iris codes removes the local dependency, but it asserts that shuffling does not completely protect against fuzzy commitment attacks because permutation is a linear, invertible transform.

Other systems allow classification to happen on the server by running classifiers on encrypted or transformed data. For example, "Blind Authentication" [20] is a general purpose authentication protocol based on homomorphic encryption. Though the work is intended for many biometric modalities, it implements an SVM-based iris classification system. The performance of the iris SVM is weak, with true-accept rates of roughly 65% at 0.1% FAR.

Some research focuses on using fuzzy vaults to bind the cryptographic key to a set of points on an over-determined polynomial. Many approaches on fuzzy vaults use multiple biometrics to improve accuracy, such as [11]. For their iris-only vault, Iris GAR = 88% when FAR $\approx$ 0.01% with 41 bits of security; the multibiometric vault had much higher performance. Some approaches such as [17] use minutiae points on structures from four quadrants of the iris. The resulting four vaults are then hardened with user passwords. They report roughly 87.2% TAR at 0.26% FAR. Using non-bit features, [8] performs ICA on "clustered" iris blocks to extract features, binding them with a 128-bit key with an accuracy of TAR=99.225% when FAR=0% on the BERC dataset. Unfortunately, techniques based on fuzzy vaults tend to have multiple unresolved security issues [19]; thus, while the above have some reasonable accuracy, they are not as secure/private as originally suggested. The proposed vaulted verification technique is similar to the concept of fuzzy vaults, but it addresses the known security concerns.

3. Vaulted Verification

Our contribution is a generalization of the Vaulted Verification protocol, adapted for iris recognition. Vaulted Verification was first introduced in [21] as a means for remote authentication between a client (C) claiming an identity and a server (B) verifying the client's claim. Vaulted Verification enables the server to ensure authentication using the client's biometric in a privacy-preserving way. While [21] presented only a reference face verifier using SVM classifiers, we will show that the approach is adaptable to other modalities such as iris codes.

The core concept behind vaulted verification is a mixture of multiple layers including the "chaffing and winnowing" technique described in [18] and the polynomial reconstruction technique from [6]. The main idea is to split a biometric sample into several parts, forcing the client to discriminate between the real and artificial chaff parts. To create a template, the enroller starts with a real feature vector from the subject and generates a chaff feature vector. The enroller then chops each vector into several "blocklets". These real and chaff blocklets are grouped into blocks; each block of the template associates one real blocklet to its corresponding chaff blocklet. From an attacker's point of view, the real and chaff blocklets look alike, so the attacker cannot tell the real from the chaff. From the client's point of view, the client can use their biometric data to decide which blocklets are real and which are chaff.

Using blocks in this way enables the server to perform a challenge-response authentication, wherein the client performs the matching process and can return a value based on the challenge to prove its result. To authenticate a potential client, the server creates a random challenge-response bitstring of n bits, where n is the number of blocks in the template. The server then creates a new, "scrambled" template: if the ith bit of the bitstring is 1, the server swaps the order of the real and chaff blocklets in the ith block of the template. If the ith bit is 0, the server leaves the ith block alone. Thus, assuming a random distribution of bits, approximately $n/2$ blocks in the template will be swapped. The server then sends this swapped template—now acting as the challenge—back to the client. It's the client's job to use that template to find the challenge-response bitstring. The client looks at each block of the swapped template and decides which blocklet is real and which is chaff by comparing each blocklet to the subject's biometric sample. Because

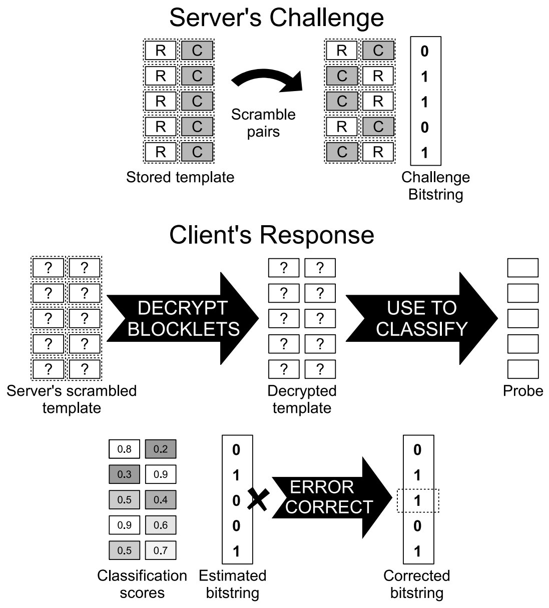

Figure 1. Overview of a simplified version of the vaulted verification authentication process. The server creates a challenge bitstring and swaps some blocks in the template, sending the swapped template to the client. The client compares each blocklet in the swapped template against a probe, using the similarity scores and error correction to recover the challenge bitstring. See Section 3.1 for details.

the client knows how the template was initially constructed, it can find which blocks have been swapped and which have not; thus, it can recover the bits of the bitstring and can send a hash of these bits back to the server, proving its identity. See Fig. 1 for an overview of the authentication process.

Because the server is only changing the *structure* of the template and is not changing its *contents*, the server does not need access to the encrypted content of actual blocklets, nor does it need to concern itself with the classification strategy—the client can use any modality or classifier to prove its identity to the server. This also means that the client can encrypt each blocklet in the template with a key that the server does not have, thus protecting the biometric features from attackers and malicious server operators.

This is the gist of the Vaulted Verification concept. Note that the actual protocol differs from the overview presented above in a number of subtle but important ways that are beyond the scope of this paper. In the remainder, we examine the iris biometric aspects of the protocol and take a detailed look at the enrollment and authentication process.

3.1. The Enrollment Process

For our experiments, we used Libor Masek's widespread open-source MATLAB iris recognition system [10]. While our approach would work with more effective techniques, use of this open-source solution will aid in comparison and reproducibility of this research.

Masek's system uses a Hough transform to segment the boundaries between the pupil, iris, and sclera. The image is transformed into a 2D rectangular Cartesian representation using Daugman's rubber sheet model. The eyelids, specular reflections, and eyelashes are segmented into a mask, and the iris code is created by quantizing real and imaginary responses to 1D Log-Gabor wavelets. The final result is a 2D iris code and a 2D mask.We used suggested parameters of a radial resolution of 20 pixels and an angular resolution of 240 pixels yielding a 9600 bits iris code. We used one filter with center wavelength = 18px and filter bandwidth $\frac{\sigma}{f}$=0.5. These parameters are close to optimal for the CASIAv1 dataset using Masek's algorithm [10, 12]. To compare two templates, Masek's system compares the hamming distances of the two iris codes, only considering the bits that are contained within both masks. Masek's final baseline classification score is the ratio of the hamming distance to the number of unmasked bits.

In our vaulted verification system, our first task, given an enrollment iris sample using Masek's system, is to generate a list of real blocklets for the template. Let B_i be the ith bit in the 1-dimensional raveled iris code (concatenating each row), and let M_i be the ith bit in the raveled mask. Our task is to split B and M into n blocklets. One naïve approach is to simply split B and M into n pieces, storing $\{(B_j, M_j) \mid 9600(x-1)/n \leq j < 9600x/n\}$ inside the xth blocklet. This is simple to implement, but has the disadvantage that each blocklet may contain a different number of useful bits. Generally, the mask contains sections of consecutive masked bits due to the large eyelids; thus, some blocklets often wind up with no useful bits at all. To prevent this from happening, each blocklet in the proposed system stores only unmasked (useful) bits along with a mapping that relates the index of each bit to its original place in the image. If B_i is the ith bit in the raveled code, then the code is transformed into $C = \{(B_i, i) \mid 0 \leq i < 9600, M_i = 1\}$; ie. the unmasked bits and their raveled indices. This new code/mapping is then permuted (randomly shuffled) to create C'. The shuffling order is determined by a mixture of a device-specific key and the user's passphrase. This does three things: First, it ensures that each blocklet does not refer to contiguous parts of the iris code, thus avoiding the contiguous masked (eyelid) region problem when matching against the probe. Second, shuffling removes the local dependency—important because consecutive bits of iris codes are not uniformly independently distributed [7]. Third, shuffling ensures that each generated template has vastly different data, thus helping to remove linkability among different encrypted templates of the same subject.

The final permuted iris code/map, C', is then chopped into n blocklets. Let k be the length of C' (that is, the number of unmasked bits), so each blocklet has $\frac{k}{n}$ useful bits of the code. Thus, blocklet $R_x = \{C'_i \mid x\frac{k}{n} < i < (x+1)\frac{k}{n}\}$. Each of these n blocklets comprise the real parts of the tem-

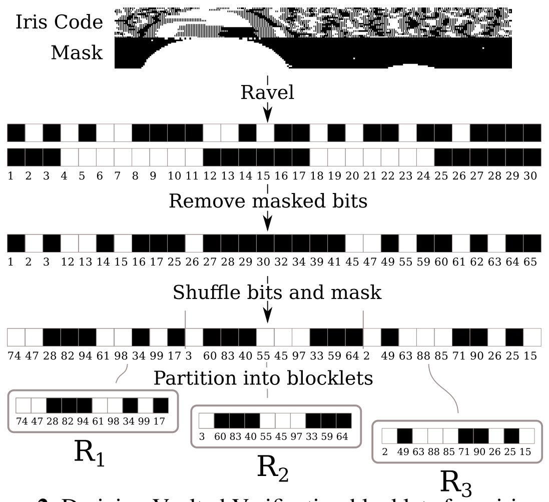

Figure 2. Deriving Vaulted Verification blocklets from iris codes. The code and mask is raveled, and masked bits are deleted. The resulting code and the indices are randomly permuted and split into blocklets.

plate. See Fig. 2 for a visual depiction of how we turn iris codes into template blocklets.

For each real blocklet R_x, we must create a corresponding chaff blocklet C_x. The original algorithm [21] randomly chose chaff from a fixed pool of chaff subjects. This had some disadvantages; namely, the need to create and store a large chaff pool for the enroller and concerns about the non-uniqueness of chaff samples. Thus, we use artificial chaff in our proposed scheme. To derive the corresponding chaff blocklet C_x for R_x, we copy the map from R_x and use uniformly distributed random bits. Because half of the bits between any two templates match on average, [2], an attacker cannot tell the difference between an R_x and C_x.

So far, our template contains a list of blocks with real and chaff blocklets (R_x, C_x). Each blocklet is then infused with error correction information. The original scheme [21] stored a point on f inside each blocklet, where f was an over-determined polynomial of degree d. Each chaff blocklet contained a random point. A hash of f's coefficients, $\textsc{Hash}(f)$, was also stored inside the template. If the client correctly guessed $d + 1$ points in the template, it could use a "homing" error correcting search to recover the coefficients and thus recover every block. In our iris experiments, we opt to use a more conventional forward error-correction strategy based on 8-bit Reed-Solomon (RS) codes. Let p be a parity parameter fixed at enrollment time. We then construct $n - p$ random symbols $S_i \in \{1, 2, \ldots 255\}$ (the "payload") which is then encoded with 8-bit Reed-Solomon encoding to create a message, D, with n symbols. This way, we can correct up to $\frac{p}{2}$ errors in a corrupted D'. We then embed each symbol of D in each real blocklet, and we embed a random symbol, different from D_i, in the ith chaff blocklet in the template. The template now contains the list of blocklets and $\textsc{Hash}(D)$, for verification.

Each blocklet is then individually encrypted with a client-specific private key as in [21]. The per-blocklet encryption protects the client's privacy from the server. Once this is done, the enroller performs one last step before storing the template. As it stands, each block contains its real blocklet first. This means that if an attacker acquires the template, they would be able to tell which encrypted blocklets are real and which are chaff, even though they could not decode the contents. To avoid this, the client performs a scrambling operation. Let S be an n-bit random key, chosen by the client (e.g. related to C''s permutation key and the symbols in D). For each block, if S_i is 1, then the client swaps the encrypted real and chaff blocklets inside block i.

This final template contains $\textsc{Hash}(D)$ along with n blocks containing swapped blocklets, each blocklet element being separately encrypted. Each blocklet either contains D_i and parts of the iris code with indices, or it contains a random message symbol and the bits and indices from part of a randomly generated iris. To prepare for storage, the enroller encrypts this final template with the server's public key and stores it on the client's device. The trusted enroller then computes a hash of the template, sending only this hash to the server for storage. In this way, the server *does not* store any templates. Except during a transaction, the client's template is never available to the server and an attacker cannot recover the template without access to the client's device and the server's private key. The client cannot modify the template because it would no longer match the server's stored hash.

3.2. Authentication process

To match a template, the client initiates authentication with the server by sending its encrypted template and the client's public key. The template was encrypted with the server's public key at enrollment, so the server verifies it with the hash in its template database and decrypts the overall template (but cannot decrypt the individual blocklets).

From this point, we can split the process into the server's role, to generate the challenge-response token and transform the template, and the client's role, to respond with the correct token by detecting the transformation.

The server starts by picking a random, session-local string (call it *nonce*). The server then creates a challenge bitstring of n bits, call it C. The server uses C to create a transaction-specific swapped template. As stated previously, if C_i is 1, the server will swap the real and chaff blocklets in the ith block; if C_i is 0, the ith block will not be swapped. The server builds a message containing the swapped blocks and $nonce$, encrypting the message with the client's public key. The server sends this encrypted message to the client and waits for it to respond. The server is expecting a response of

$$\textsc{Hash}(\textsc{Hash}(D) \mid\mid nonce \mid\mid C). \quad (1)$$

where $||$ denotes concatenation. The server has all the pieces necessary to validate the client's answer – $nonce$ and C are chosen at the beginning of the session, and $\textsc{Hash}(D)$ is stored in the template. The client must recover $nonce$, C and $\textsc{Hash}(D)$ using its private key, biometric and the ECC data. The server indicates a positive authentication if and only if the client's response equals the expected response.

The client's role is marginally more complex. First, the client decrypts the message to recover $nonce$ and the swapped blocks. It must then find C using its probe biometric sample. To decide which blocklet in each block is real or chaff, the client decrypts each blocklet, yielding bits T and the mapping N. Let B' be the 9600 bits in the probe biometric sample, and let M' be the probe's mask. Each blocklet's similarity score is

$$\text{score} = 1 - \frac{\|(T_x \oplus B'_{N_x}) \wedge M'_{N_x}\|}{\|M'_{N_x}\|} \qquad (2)$$

where $\|\alpha_x\|$ denotes the number of 1 bits in α for $x < \textsc{Length}(\alpha)$. This way, the client finds scores for both blocklets in each block. The blocklet with the higher score is assumed to be the real blocklet, and the lower-scoring blocklet is assumed to be chaff.

Once the client has these choices, it constructs D' using the message symbols inside the suspected real blocklets. As long as $n - \frac{p}{2}$ guesses are correct, the client can decode the answer using RS decoding to retrieve D. This process reveals the actual symbols in the real message, thus showing the client which blocklets are real and which are chaff. From there, the client can use their knowledge of S to derive the swapping used in the original template, and can trivially compute C and $\textsc{Hash}(D)$. Iris images tend to have a small amount of variable rotation. As such, the client performs 16 trials per probe, shifting B' left by up to 8 bits and right by up to 8 bits. This yields 16 lists of scores per block per rotation. The client can choose the rotation that unlocks D.

4. Security

To show how Vaulted Verification resists attacks of varying sophistication, we start by examining the security of the scheme from an attacker's point of view.

If an attacker has no encryption keys but has a copy of the stored template, they cannot acquire or discern the subject's iris codes. To show this, we see that by default, the attacker cannot discern any of the iris codes contained in the blocklets because they are encrypted with the client's key and the template is encrypted with the server's key. Because we assume the attacker cannot decrypt a ciphertext without possessing the key, the subject's iris code is secure.

The server (or an attacker) also cannot run an automated verification attack because they do not have enough information to perform a classification; they cannot compare the encrypted blocklets to an arbitrary probe.

Finally, the attacker cannot authenticate as a client. To see why, recall that our protocol requires the attacker to present $\textsc{Hash}(\textsc{Hash}(D)||nonce||C)$. We know that $\textsc{Hash}(D)$ is encrypted in the template with the server's key; it is irrecoverable. C is derived only from the canonical swapping S and knowledge of which blocks are real and chaff in each pair. If we assume the attacker does not have D, S, or the client's key, it cannot find C. Further, without a probe sample, the attacker's chances of guessing C is equal to $\frac{1}{2^N}$, and the attacker cannot verify his answer without sending it to the server and waiting for an authentication decision. An attacker cannot begin an authentication attempt with the server because it does not have the encrypted template. Thus, by default, security is preserved.

4.1. Man-in-the-Middle and Replay Attacks

Having shown our system's security in the default case, we now gradually give the attacker more information, evaluating the security of our algorithm at each step.

Throughout the authentication process, we assume that the client and server communicate within a properly implemented encryption protocol (SSL or TLS, for example) such that the client can verify the server's identity.

If an attacker gains the ability to subvert the network connection between the client and server by monitoring or impersonating a server or staging a man-in-the-middle attack, the attacker still cannot acquire the subject's iris code, they cannot successfully impersonate the server without the client's knowledge, and they cannot gain any information that helps in future authentication processes.

To see why, consider that when the attacker mounts their network attack, they can see the encrypted template as it first passes from the client to the server. This does not help them because the template is still encrypted with the server's public key. The man-in-the-middle attacker could record each encrypted blocklet in the swapped template along with the session-local $nonce$, and then pass these to the client. Because C and $nonce$ is random for each session, this knowledge does not foster a replay attack since the template is swapped differently from session to session. However, because only the ordering of the encrypted blocklets changes, the attacker can now watch two transactions and see whether they have the same encrypted blocklets; thus, an attacker can tell when two authentication attempts try to claim the same identity producing a weak linkability.

After the server sends the transformed template, the client's only response is $\textsc{Hash}(\textsc{Hash}(D)||nonce||C)$. Assuming that $\textsc{Hash}$ is a one-way function and assuming that $nonce$ and C are unique to each session, the attacker cannot learn anything about D or C from the client's response; thus, all information the attacker receives is not useful in future responses.

Finally, the attacker cannot pose as the server because it cannot decrypt the template at the start of the protocol without the server's private key.

4.2. Server Encryption and Client Storage

To decrypt the template, the attacker needs access to both the server's private key and the client's device. We expect that only highly determined attackers will get this far, but what information do they gain if they have the server's key and the template? For example, what if the attacker has compromised the server? Recall that the template contains $\textsc{Hash}(D)$ and the encrypted blocklets. At this point, the attacker is capable of authenticating as the client. This is because the client's response is $\textsc{Hash}(\textsc{Hash}(D)||nonce||C)$. The attacker has $\textsc{Hash}(D)$ because it is stored in the template. The attacker can find C because it's implicit in the change of structure between the original template and the swapped template sent by the server. The server picks $nonce$, so it would be known if the server is compromised.

The attacker can now authenticate as the subject for this particular server, but he or she still cannot recover the subject's original iris code or impersonate them on another server. He/she does not have the ability to decrypt the blocklets because they are encrypted with the client's private key. The attacker also does not know which are real and which are chaff because S is derived from the client's keys and/or passphrase. If there was reason to believe the server was compromised, the template can be revoked and reissued.

4.3. Client Encryption

What can the attacker gain if they acquire the ability to completely remove all encryption by stealing the template, the server's key, and the client's keys? At this point, the attacker knows the decrypted blocklets but cannot discern between the real and chaff because he or she does not know S. Thus, the biometric is still obscured, but only by the scrambling and the presence of chaff. If the attacker wishes to find which blocks comprise real and which blocks comprise chaff, he or she now has the ability to launch an offline brute-force attack. This attack can be performed without any intervention or knowledge from the client or the server. To find the real and the chaff, the attacker constructs D' by picking one symbol from all n blocks of the template. The attacker can then try to decode D using the Reed-Solomon decoding process. If D' decodes, the attacker can verify their guess by comparing $\textsc{Hash}(\text{RS}\textsc{Decode}(D'))$ against $\textsc{Hash}(D)$. Once the attacker has the symbols that comprise D, the attacker knows know which blocks are real, and can extract the iris code and mapping.

With this strategy, an attacker must make, on average, $2^{n-p/2-1}$ guesses before they correctly find D. For a template with 128 blocks and $p = 54$ capable of correcting 27 errors, the attacker must enumerate an average of 2^{100} possible combinations of symbols in a brute force attack. This is not impossible with today's computing power, but recall that to get to this level, we have already presumed the attacker has gained physical access to the client's device and obtained the client's passphrase to retrieve the client's private key. The attacker has also compromised the server or acquired a copy of the encrypted template along with the server's private key. If we move to 256 blocks, we can make this brute force attack impractical for decades to come.

5. Evaluation

Accuracy is a critical security concern. How well does our iris verification system perform? To evaluate the performance of our approach, we tested our proposed system on CASIAv1. Masek [10] used a custom subset of CASIAv1, dubbed CASIA-a, which contained the 624 well-segmented, manually checked irises correctly found by his segmentation algorithm. Masek's segmentation algorithm has about 83% accuracy. Because we do not know which images from CASIAv1 comprise CASIA-a, we used our own subset in these experiments, dubbed CASIA-b, that's substantially similar to CASIA-a but with 655 images from 106 subjects. Like Masek's approach, we manually verified the segmented irises, but since our database includes 24 more images, we cannot directly compare our results with Masek's originally reported performance. However, since our goal is understanding Vaulted Verification's impact on the accuracy, and because we are using Masek's original code, it's not critical that we exactly match that performance. What will matter is the change in performance as we add privacy protection.

The experimental protocol dictates that all possible image-image comparisons in the database should be taken. Two images that are taken from the same subject count as a true accept or false reject, and two images from different subjects count as a true reject or false accept, depending on the algorithm's decision. Using the open-source implementation of Masek's algorithm yields a true accept rate of 92.056% at 0.100% FAR on CASIA-b. It is well-known that segmentation has a large impact on iris verification performance [12], particularly for verification problems because the worst-quality images in the database hurt the true accept rate. Improved segmentation for CASIA-b would likely improve performance as we only removed the obviously failed segmentations. The distribution of our "Masek baseline" is shown at the top of Figure 4.

It's important to note that in our experiments, we assume that the attacker has access to all user keys and passwords—these results show Vaulted Verification's ability to withstand a stolen token attack. If the attacker does not have access to the subject's passwords and other authentication data, we expect the false accept rate to be 0 because the attacker cannot decrypt each blocklet and thus has a minimal chance of guessing C and $\textsc{Hash}(D)$.

Though Vaulted Verification is general enough to apply to many different biometric modalities, it makes the as-

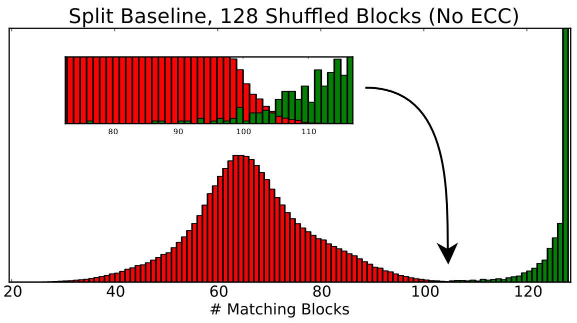

Figure 3. Distribution of match scores in the "baseline" split experiment with no ECC on CASIA-b, along with a close-up of the area between impostor and subject scores.

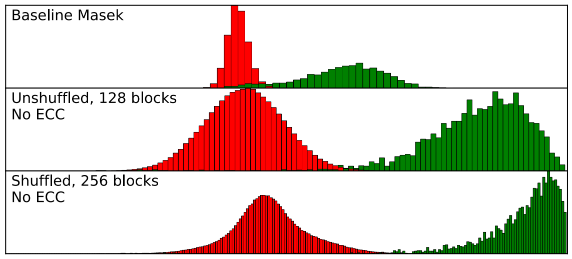

Figure 4. Score distributions on CASIA-b. Masek baseline is on top. When compared to Figure 3, we see that shuffling (middle) and splitting the iris code into more albeit smaller blocks (bottom) has a large impact on the distribution.

sumption that splitting the template into several blocks does not hurt performance. To measure the impact of this assumption, we conducted a "baseline split" experiment by converting irises into 128 blocks as described in Section 3.1. The final score is the number of real blocklets in the template that match the probe. No error correction information is used, and no challenge/response is performed. Thus, this experiment does not test Vaulted Verification's performance; rather, it tests how the block-splitting assumption impacts iris verification performance in general. From Figure 3, we see that the impostor scores are centered at 64 matching blocks (as expected), but the vast majority of the subject templates match all 128 blocks. This is a vastly different distribution than most iris recognition systems, and we believe it happens this way for two reasons: First, consolidating 7424 bits (the average number of unmasked bits across CASIA-b) into 128 blocks tends to exaggerate the difference between real and chaff scores. An attacker, on average, will only match 50% of the bits and thus 50% of the blocks, but the honest subject now has many more opportunities per block to distinguish between real and chaff because each block now contains up to $\frac{7424}{128} = 58$ good bits. Second, randomly permuting the code distributes more useful data to each block. Without this shuffling, all of the bits inside each block may fall within the probe's eyelid or in other large scale differences. This experiment outperformed Masek's baseline with 98.889% TAR at 0.138% FAR on CASIA-b. For reproducibility, we note that on all of

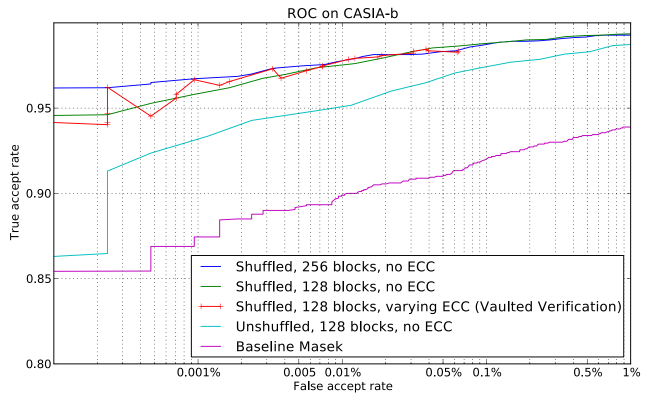

Figure 5. ROC of various algorithms on CASIA-b.

CASIAv1, including badly segmented irises, our algorithm performed at 90.234% TAR at 0.119% FAR.

To test this reasoning, we ran experiments without the shuffling step; see Figure 4 (middle). As expected, performance of this baseline without shuffling decreases when compared to the shuffled performance, with TAR=97.695% at FAR=0.148%. We additionally tried shuffling but with a smaller blocklet size; at 256 blocks, the number of good bits per blocklet falls to roughly 29 (CASIA-b has 7424 unmasked bits per sample on average, divided by 256 blocklets). The performance is 98.861% TAR at 0.12% FAR, comparable to the shuffled 128-block template. Figure 4 (bottom) shows the resulting distribution. Note that both of these changes alter the genuine score distribution when compared with Masek's algorithm. We see that even without shuffling the code, consolidating 7424 bits into 128 blocks increases the distance between subject and impostor scores compared to our Masek baseline, and this distance is exaggerated further when the shuffling step is performed in Figure 3. When we increase the number of blocks to 256, we see that the score distance between real and chaff is altered, but the performance is not significantly reduced because the tail of the distributions has a greater impact on performance than the mean of the distributions. Note that a similar separation phenomenon due to shuffling was observed in [7]. The resulting ROC of these experiments is shown on Figure 5.

Finally, we perform the Vaulted Verification experiment several times, varying the ECC parameter and showing the resulting ROC in 5. This experiment assumes the attacker has all keys and only tests the biometric's ability to secure the ECC data in the template. Note that this curve on the ROC is not monotonic because at such low FAR rates, the different random permutations of C have a great effect on the tail of the impostor's scores, which vary between tests. We varied the ECC parity as the threshold; for example, at $p = 30$, subjects and impostors can correct up to 15 incorrect bits for a total security of $128 - 15 = 113$ bits. At higher FAR, this closely approximates our 128 shuffled block baseline. On CASIA-b at these parameters, our algorithm performs at 94.02% TAR when FAR=0.0002% at 115 bits of security, and 98.22% when FAR=0.0310% at 105 bits of security. For reproducibility, we note that on CASIAv1 including badly segmented irises, for 115 bits of security, FAR=0 and TAR=81.72%; for 108 bits of security, TAR=87.80% and FAR=0.0125%. Using error correction in this way allows us to guarantee that the client can decode all the blocks in the template at the given level of ECC. This has advantages in the protocol as discussed in Section 4. We did not perform ECC on the shuffled 256-block experiment because 128-bit ECC did not significantly impact performance; we would expect similar performance from 256.

6. Conclusions and Future Work

In this work, we have shown how to create an iris verification system adapted to the unique privacy and security requirements of remote authentication. We showed Vaulted Verification's privacy and security and demonstrated its accuracy on the CASIAv1 iris database. We found that Vaulted Verification outperformed the baseline when using 128 shuffled blocks. In the future, we wish to find the optimal template block count and to explore the accuracy of the system across many different iris feature extraction algorithms and better iris segmentation algorithms. In addition, we also wish to investigate performance on a variety of Reed-Solomon parameters. We also want to investigate how Vaulted Verification can be used for liveness detection, and how to adapt Vaulted Verification to more biometric modalities.

Acknowledgments

Portions of the research in this paper use the CASIA-IrisV1 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA). The research reported in this document has been funded partially by NSF grants CNS-0958576 and CNS-0851783.

References

[1] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, and G. Zemor. Theoretical and practical boundaries of binary secure sketches. *IEEE Trans. Inf. Forens. Security*, 3(4), 2008.

[2] J. Daugman. The importance of being random: statistical principles of iris recognition. *Pattern Recognition*, 36(2), 2003.

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

[4] F. Hernndez lvarez, L. Hernndez Encinas, and C. Snchez vila. Biometric fuzzy extractor scheme for iris templates. In *WORLDCOMP*, 2009.

[5] D. Jeong, H.-A. Park, K. Park, and J. Kim. Iris recognition in mobile phone based on adaptive gabor filter. In D. Zhang and A. Jain, editors, *Advances in Biometrics*, LNCS # 3832, pages 457–463. Springer Berlin / Heidelberg, 2005.

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

[7] S. Kanade, D. Petrovska-Delacretaz, and B. Dorizzi. Cancelable iris biometrics and using error correcting codes to reduce variability in biometric data. In *CVPR*, june 2009.

[8] Y. Lee, K. Bae, S. Lee, K. Park, and J. Kim. Biometric key binding: Fuzzy vault based on iris images. In *Advances in Biometrics*, LNCS # 4642, pages 800–808. Springer Berlin / Heidelberg, 2007.

[9] 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.

[10] L. Masek. Recognition of human iris patterns for biometric identification. Master's thesis, University of Western Australia, 2009.

[11] K. Nandakumar and A. Jain. Multibiometric template security using fuzzy vault. In *BTAS*, 2008.

[12] T. Peters. Effects of segmentation routine and acquisition environment on iris recognition. Master's thesis, University of Notre Dame, 2009.

[13] J. Pillai, V. Patel, R. Chellappa, and N. Ratha. Sectored random projections for cancelable iris biometrics. In *IEEE Acoust., Speech, Signal Process.*, 2010.

[14] J. Pillai, V. Patel, R. Chellappa, and N. Ratha. Secure and robust iris recognition using random projections and sparse representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(9), 2011.

[15] E. Pschernig. Cancelable biometrics for iris detection with parameterized wavelets and wavelet packets. Master's thesis, Universität Salzburg, 2009.

[16] C. Rathgeb and A. Uhl. Statistical attack against iris-biometric fuzzy commitment schemes. In *CVPR Workshops*, 2011.

[17] E. Reddy and I. Babu. Performance of iris based hard fuzzy vault. In *IEEE Int. Conf. Comput. Inform. Tech. Workshops*, 2008.

[18] R. L. Rivest. Chaffing and winnowing: Confidentiality without encryption. Technical report, MIT Lab for Computer Science, 1998.

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

[20] M. Upmanyu, A. Namboodiri, K. Srinathan, and C. Jawahar. Blind authentication: A secure crypto-biometric verification protocol. *IEEE Trans. Inf. Forens. Security*, 5(2), 2010.

[21] M. J. Wilber and T. E. Boult. Secure remote matching with privacy: Scrambled support vector vaulted verification (s2v3). In *WACV*, 2012.

[22] S. Yang and I. Verbauwhede. Secure iris verification. In *IEEE Acoust., Speech, Signal Process.*, volume 2, 2007.

[23] X. Zhou, A. Kuijper, and C. Busch. Retrieving secrets from iris fuzzy commitment. In *ICB*, 2012.

[24] J. Zuo, N. Ratha, and J. Connell. Cancelable iris biometric. In *ICPR*, 2008.