Reusable Authentication from the Iris

Sailesh Simhadri, James Steel, and Benjamin Fuller

University of Connecticut
Storrs, CT 06269
Email: `first.last@uconn.edu`

December 1, 2017

Abstract

Mobile platforms use biometrics for authentication. Unfortunately, biometrics exhibit noise between repeated readings. Due to the noise, biometrics are stored in plaintext, so device compromise completely reveals the user's biometric value.

To limit privacy violations, one can use fuzzy extractors to derive a stable cryptographic key from biometrics (Dodis et al., Eurocrypt 2004). Unfortunately, fuzzy extractors have not seen wide deployment due to insufficient security guarantees. Current fuzzy extractors provide no security for real biometric sources and no security if a user enrolls the same biometric with multiple devices or providers.

Previous work claims key derivation systems from the iris but only under weak adversary models. In particular, no known construction securely handles the case of multiple enrollments. Canetti et al. (Eurocrypt 2016) proposed a new fuzzy extractor called *sample-then-lock*.

We construct biometric key derivation for the iris starting from sample-then-lock. Achieving satisfactory parameters requires modifying and coupling of the image processing and the cryptography. Our construction is implemented in Python and being open-sourced. Our system has the following novel features:

1. 45 bits of security. This bound is pessimistic, assuming the adversary can sample strings distributed according to the iris in constant time. Such an algorithm is not known.
2. Secure enrollment with multiple services.
3. Natural incorporation of a password, enabling multifactor authentication. The structure of the construction allows the overall security to be sum of the security of each factor (increasing security to 79 bits).

1 Introduction

Biometric Authentication Authentication allows devices to identify human users. Passwords are the dominant authentication paradigm on desktop computing platforms. Bonneau et al. provide an overview in their systemization of knowledge [BHVOS12]. Biometrics are preferred on mobile platforms including phones, tablets, and wearables. Prominent mobile examples include fingerprints [WJMM05], irises [Dau04], and facial geometry [BS00]. Biometrics are often adopted for usability reasons but there are important security and privacy considerations. In this work, we focus on the human iris which is believed to be the strongest biometric [PPJ03]. Thus, if we are unable to provide strong privacy and security for the iris, prospects look bleak for other modalities. Two challenges in biometric authentication are 1) biometrics exhibit noise between repeated readings, necessitating a matching algorithm that allows for error tolerance and 2) biometrics cannot be regenerated or refreshed.

Noise Consider some initial iris value w. At authentication time a person presents a subsequent reading w'. The system compares w' to w. The natural way of performing this comparison is to store w, but storing w creates security and privacy risks. Password systems mitigate this risk by storing a salted hash of the password instead the password. In biometric systems because of physiological, environmental, and sensor noise the value w' is extremely unlikely to be the same as w. Instead, w and w' are close according to some distance metric, that is, $\mathsf{dis}(w, w') \leq t$ for some parameter t. However, cryptographic hashes are supposed to map two different values to "independent" outputs. It is unclear how to adapt the approach of storing a hash of the original reading. While there are hashes that are intended to preserve distance, known as locality sensitive hashing [PJA10], we do not have such hashes for the distance metrics that occur in biometrics. Thus, biometric authentication systems store w, creating two major weaknesses:

1. Need for private storage and trusted computing base. Biometric compromises are more damaging than password compromises as a user cannot create a new value.

2. Servers can check which users they share. This is a violation of the desirable property of *unlinkability.* In addition, a malicious server can completely impersonate their enrolled users.

Key derivation from noisy sources An alternate approach derives stable cryptographic keys from noisy data. A *fuzzy extractor* [DRS, DORS08] is the cryptographic primitive that performs this task. Fuzzy extractors consist of a pair of algorithms: Gen (used at enrollment) takes w, and produces a key r and a public helper value p. The second algorithm Rep (used at authentication) takes this helper value p and a close w' to reproduce the original key r. In theory, fuzzy extractors address drawback 1. However, Blanton and Hudelson note that standard fuzzy extractors are not secure for actual biometrics including the iris [BH09, Section 5]. Furthermore, fuzzy extractors provide no security guarantee if a user enrolls their biometric with multiple servers. Secure reuse is more important for biometrics than passwords [BHVOS12, Pg. 564].

Reusability A fuzzy extractor is *reusable* (Boyen [Boy04]) if it remains secure even when a user enrolls their biometric with multiple different servers. Security is defined as follows, each server i gets a different enrollment reading, denoted w_i, and runs $\mathsf{Gen}(w_i)$ to get a key r_i and a helper value p_i. Each r_i should be a cryptographic key even if an adversary is given all the values $p_1, \ldots, p_\rho$ and the keys r_j for $j \neq i$. Reusability implies that the produced r are unlinkable mitigating drawback 2 (looking ahead, the first two versions of our scheme are unlinkable knowing both r and p). Deployed constructions of fuzzy extractors are not reusable [Boy04, STP09, BA12, BA13] and there are no known constructions that meet the correctness and security needs of biometrics.

1.1 Our Contribution

In this work we construct a reusable key derivation for the human iris. Our construction builds on the recent construction of Canetti et al. [CFP+16] called *sample-then-lock.* The idea is to sample a random subset of the processed iris, hash these bits, and use the output of the hash as a pad for cryptographic key. This process is repeated many times with the same key. The intuition is that one of these subsets should have no errors allowing recovery of the key.

The parameters of sample-then-lock require an error rate that is sublinear in the length of output of the iris transform. Processing two images of the same iris results in binary strings that differ in between $10\% - 32\%$ of bits (depending on the transform). Sample-then-lock cannot natively handle the iris. Overcoming this challenge requires modification of the iris processing and cryptography. Our construction feeds additional information from iris processing to the fuzzy extractor coupling the two stages. We highlight the most important features of our system below.

Security The lower bound for security of our system is 45 bits. However, this bound is conservative, assuming that the adversary can in a single time unit 1) create 65 bit strings distributed in the same way as processed iris bits and 2) evaluate a cryptographic hashes. Due to the extensive investment in password cracking, evaluating hashes is very fast. However, we are not aware of any fast method to sample strings distributed according to the processed iris. We believe this operation is currently difficult for the iris but there could be algorithmic advances. Thus, we believe the actual security of our method is higher than 45 bits.

Scheme	Limitation
Code Offset [Boy04]	Assumes noise uncorrelated to iris
LWE Decoding [ACEK17]	Assumes noise uncorrelated to iris
Pseudo. isometry [ABC+16]	Applicable for set difference metric
Sample-then-lock [CFP+16]	Sublinear correction capacity

Table 1: Recent constructions of reusable fuzzy extractors. The code offset and LWE decoding schemes leak information about the difference between repeated readings w_i, w_j which may be correlated to the individual readings.

Reusability Our construction is a reusable fuzzy extractor allowing a user to enroll with multiple service providers and know that their security is retained if all but one service provider is adversarial. Importantly, the security argument makes no assumption about how repeated readings of the iris are correlated. As we discuss below, this has been a limitation of prior work.

Multifactor system Many previous systems combine biometric authentication with a password/passcode [sam]. These systems treat the biometric and password separately, checking if w' is close enough and separately if the hashes of the passwords match. Our system consists of repeated hash evaluation naturally allowing a password to be added to each hash. This allows us to achieve security which is the sum of the security of the two methods. An 8 character password with a mix of upper and lower case, symbols, and numbers is estimated to have 34 bits of entropy [KSK+11].

Our construction has been implemented in a prototype Python implementation. On average Gen takes 220s and Rep takes 12s. We expect both of these numbers would decrease by an order of magnitude using a lower level implementation. In our evaluation we targeted a correctness rate of 50% with measured correctness higher at 59%. Correctness can be increased by small parameter changes, which we discuss in Section 2. All of our analysis is based off the ND-0405 iris dataset [BF16].

On security of 45 bits We are aware that 45 is not "cryptographically secure." Adding a block to the Bitcoin blockchain requires $\approx 2^{40}$ hashes and is done every 10 minutes. We believe use of this system increases security and privacy of individuals against many threats. As a comparison, passwords are regularly hashed and salted in both server and desktop environments despite having lower security than our system. Lastly, in the *sample-then-lock* construction the security of the system is directly tied to the error rate of the biometric. Our implementation was built on top of an open-source iris processing library (OSIRIS [KMSD17]), we expect a state of the art iris processing library to exhibit a lower error rate and thus higher security.

1.2 Prior work

We split our discussion of prior work into two parts, cryptographic theory and systems.

Reusable Fuzzy Extractors Boyen [Boy04] defined reusable fuzzy extractors in 2004 and showed that even when the exclusive OR of enrollments does not leak any information reusability requires a large decrease in security [Boy04, Theorem 11].[1] There was not much subsequent work even in this weak model [ACEK17]. Related work shows that many fuzzy extractors are not reusable [STP09, BA12, BA13], meaning that the negative result of Boyen is not only a theoretic issue.

Boyen's result is an information-theoretic negative result, recent work considers computational security [FMR13] in part to avoid this result. Alamelou et al. construct a reusable fuzzy extractor is for a different distance metric (set difference metric), their system is not applicable for the iris [ABC+16]. The only construction that appears viable is the sample-then-lock construction (our starting point). However, sample-then-lock does not support enough errors to produce a usable system. All known reusable fuzzy extractors are summarized in Table 1.

[1]The actual result of Boyen applies to *secure sketches* which imply fuzzy extractors. A secure sketch is a frequently used tool to construct a fuzzy extractor. Our construction does not utilize a secure sketch.

Iris Key Derivation The work of Hao et al. [HAD06] constructs a fuzzy extractor using an error-correcting code tailored to the iris.[2] They correct approximately 27% of errors. Importantly, the scheme of Hao et al. use a propreitary transform and dataset due to Daugman [Dau04] which has a mean error rate of 10%. In other databases, mean error rates between 25% and 30% are common. They claim a key size of 140 bits. Hao et al. implicitly consider an adversary that randomly samples an iris and asks how often this iris authenticates. This security model captures an online adversary, however a fuzzy extractor only provides better security than a plaintext matching approach when an adversary captures p and uses it to learn about viable w.

Bringer et al. [BCC$^+$07] do not explicitly describe a key length but they report a nonzero false accept rate which implies a very small effective key strength (see discussion in [ICF$^+$15]). Reporting a nonzero false accept rate is common in iris key derivation despite claimed key lengths > 40 bits (see [PRC15]). Kanade et al. [KCK$^+$08] claim a fuzzy extractor construction but they report the entropy of the iris as over 1000 bits, much higher than other estimates. Other research assumes that each bit of the iris transform is independent [GKTF16] which is not true (see for example our statistical analysis in Section 3). The above is only a sampling to this field, see the survey of Bowyer et al. [BHF13, Section 6]. There are a wide variety of claims in prior work, but 1) no prior construction has a clear definition of security and clear statement of necessary conditions for security and 2) no prior work has constructed a reusable key derivation system for the iris.

1.3 Future Work

Our system is based on repeated evaluation of a cryptographic hash. We do not consider hash functions that are difficult to compute on GPUs. There are many promising memory hard hash functions such as scrypt [PJ16] and argon2i [Jos15]. Using these constructions in sample-then-lock is nontrivial as the hash function must be computed many times by the honest party. Ideally, one could use a hash function that is easy to compute in parallel with fast access to a large memory but hard for GPUs. We are unaware of any such candidates.

Not all biometric authentication use cases allow fast password entry, e.g. unlocking a smart phone. A two factor system will require some ingenuity to be implemented in these settings. It may be possible to construct a hybrid system where both factors are required at startup but only the iris is required while the device is in use. Such hybrid systems where one authentication method is required on restart and another during use are common on smart phones.

1.4 Organization

The remainder of this work is organized as follows: in Section 2 we describe iris processing, sample-then-lock, and why they cannot be immediately combined, in Section 3 we provide background on iris code transformations and the parameters of our scheme, in Sections 4 and 5, we integrate the iris processing and sample-then-lock components into our system, in Section 6 we add a password, in Section 7 we describe our implementation and conclude in Section 8.

2 Background

The starting points for our system are the OSIRIS open source iris code transform [KMSD17] and the reusable fuzzy extractor of Canetti et al. [CFP$^+$16] called *sample-then-lock*. We briefly describe both components here.

2.1 The IrisCode Transform

The raw iris image is not used for authentication. Image processing techniques are first applied to the iris. The OSIRIS package takes a standard near infrared iris image and produces a 32768 bit vector w. The goal is to arrive at a set of features that are stable under environmental variation and image noise. Mathematically, these features should be close together under the Hamming metric (fraction of bits that differ). The stages of this process are:

[2] Most fuzzy extractor constructions use error correcting codes in some fashion. Recent work has shown it is possible to transform most coding based constructions into one another [DGV$^+$16]. Thus, we do not note how error correcting codes are used.

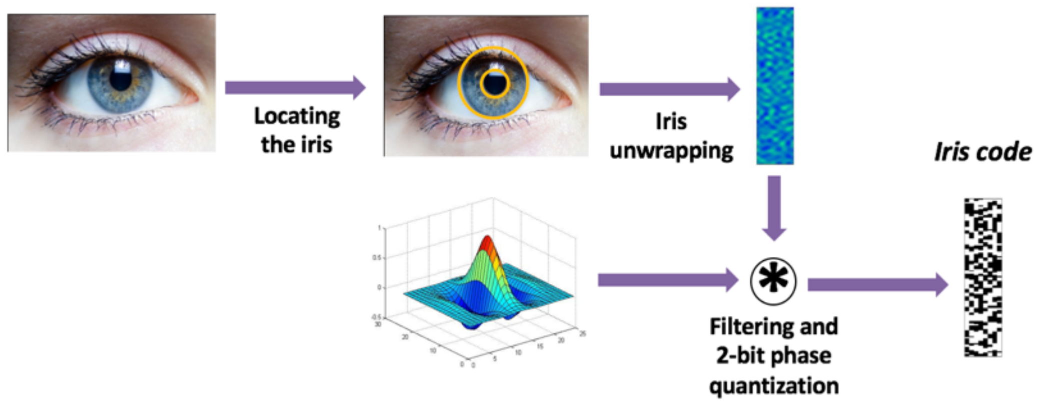


Figure 1: The major steps of most iris recognition algorithms. The final output is a polar array of bits based on the quantization of convolution with a Gabor filter.

1. Iris and Pupil Localization: This step finds the inner and outer boundaries of the iris accounting for pupil dilatation and occlusions due to the eyelid or eyelashes.
2. Iris Unwrapping: The iris is converted into a 2D matrix (using the rubber band transform). This array is indexed by (r, θ) which is the polar position of the pixel in the original image.
3. Featurization: 2D Gabor filters [GM84] centered at different positions are convolved with the image to compute a complex valued number at location (r, θ) throughout the unwrapped image. This produces a 64×512 vector of complex valued numbers.
4. Binarization: Each complex number is converted into two bits based on whether the real and imaginary components are > 0 or ≤ 0.

This process is presented visually in Figure 1. The final output of the process is a 2D matrix. Featurization occurs in a polar coordinate system and the dimensions of this matrix correspond to the number of radial r (64) and angular θ (512) positions where wavelet transforms were computed. The iris biometric has several desirable properties:

1. Different individuals have fractional Hamming distance tightly distributed around .5.
2. The features are estimated to have 249 bits of entropy [Dau04].
3. Images from the same individual have a mean fractional Hamming distance of $.11 - .32$. The lower end of this range is reported by [Dau04], in the OSIRIS system we observe a mean error rate of .32 using the ND-0405 dataset [PSJO+06, BF16].

Iris comparison are performed using the Hamming metric (the number of positions where $w \neq w'$).

2.2 Sample-then-lock

Our cryptographic construction is based on the reusable fuzzy extractor of Canetti et al.[3] A fuzzy extractor consists of two algorithms Gen which on input w produces a uniform key Key and a public string p. The second algorithm Rep on input w' and p reproduces the key Key if $d(w, w') \leq t$. The high level idea is to encrypt the same key multiple times using different subsets of w. Pseudocode for the algorithm is below:

Gen(w):

[3]In our discussion, we assume that a HMAC-SHA512 [PUB95] serves as a digital locker which is a strong form of symmetric encryption [CD08, CKVW10]. We define the relevant cryptographic primitives in Appendix A.

```
1. Sample random 256 bit Key.
2. For $i = 1, ..., \ell$:
     (i) Choose $1 \leq j_{i,1}, ..., j_{i,k} \leq |w|$
    (ii) Choose 512 bit hash key $h_i$.
   (iii) Set $v_i = j_{i,1}, ..., j_{i,k}$.
    (iv) Set $c_i = \texttt{Hash}(h_i, w_{j_{i,1}}, ..., w_{j_{i,k}})$.
     (v) Set $p_i = (0^{256} || \mathsf{Key}) \oplus c_i$.
3. Output $(\mathsf{Key}, p_i, v_i, h_i)$.
```

$\mathsf{Rep}(w', p_1, ..., p_\ell, v_1, ..., v_\ell, h_1, ..., h_\ell)$:

```
1. For $i = 1, ..., \ell$:
     (i) Set $c_i = \texttt{Hash}(h_i, w'_{j_{i,1}}, ..., w'_{j_{i,k}})$.
    (ii) If $(c_i \oplus p_i)_{1..256} = 0^{256}$ then
         output $(c_i \oplus p_i)_{257..512}$.
2. Output $\perp$.
```

In the description above, $x_{a..b}$ denotes the restriction of a vector to the bits between a and b while 0^{256} represents the 256 bit string that is all zeroes.

The parameters k and ℓ represent a tradeoff between correctness and security that we discuss shortly. The sample-then-lock construction takes multiple uniformly random samples of bits from the iriscode (these samples are of size k). These bits are then used as input to a hash function which acts as a mask for the encryption key Key. For the scheme to be secure, the hash function must function as a digital locker (see Appendix A). A digital locker leaks no information about Key as long as the input subsets $w_{j_i,1}, ..., w_{j_i,k}$ have high entropy.

For the scheme to be correct at least one of the ℓ subsets should have no error with high probability. Canetti et al. show it is possible to set ℓ if the expected error rate is sublinear in $|w|$. That is, when $d(w, w')/|w| = o(|w|)$. However, this is not the case for any known biometrics.[4]

2.3 Inadequacy of naive combination

The sample-then-lock construction is not simultaneously secure and correct for the iris:

1. The parameter k must be large enough to prevent brute force attacks. However, increasing k decreases the correctness guarantee. We observe an mean error rate of 32% using the ND-0405 Iris data set [PSJO$^+$06,BF16]. While Daugman reports mean error rates of 11%, we are unaware of any subsequent work that achieves as low an error rate as 11%.[5] If set a security target of 40 bits of entropy, this implies that $k \geq 40$. In this setting, the probability of opening a single locker is $(1 - .32)^{40} \approx 2 \times 10^{-7}$. To achieve a 50% true positive rate (solving the equation $(1 - 10^{-7})^x = .50$) requires 2×10^6 digital lockers.[6]

2. The security of the construction does not immediately following from having a large k. For security it is sufficient for subsets of w to have high entropy. It is not known how much entropy is contained in subsets of the iris code. Sampling random subsets is known to not decrease entropy rate [Vad03]. However, the entropy rate of the iris, $H(W)/|W|$, is only 1%. If the entropy rate of samples is also 1%, achieving 40 bits of effective security would require subsets of size 4000 bits requiring storage of roughly 10^{600} digital lockers.

[4]Asymptotic notation is not well defined for a single distribution W. However, we show that the 35% error rate observed in irises is not correctable by this construction.

[5]Our scheme immediately improves with an iris transform with lower error rate.

[6]Each digital locker requires storage of at least $32 + 32 + \log |w|/8 * k = 177$ bytes corresponding to the hash c_i, the salt h_i, and the selected positions, $j_{i,1}, ..., j_{i,k}$).

The rest of this work is dedicated to overcoming these two problems. Throughout our discussion we will set the number of lockers $\ell = 10^6$. This will create storage of ≈ 20 MB which is feasible for most devices. We will assume a *correctness target* of 50% true positive rate. While this number is unacceptable for an authentication system, correctness rate is an "s-curve" in error rate. Correctness increases quickly once it hits 50%, achieving correctness of $1 - 2^{-x}$ for some x requires multiplicatively increasing the number of lockers by 2^{x-1}. So 93.75% correctness requires 8×10^6 lockers. We consider these parameters fixed.

Our focus is on improving two parameters: 1) the entropy of each subset fed to the digital locker and 2) the size of each subset. We call these parameters *security* and *subset size* respectively.

The only strategy to attack this system (assuming security of the digital locker) is repeated evaluation of the hash function. *Security* measures the difficulty of "breaking" the hash for an adversary that can produce strings distributed according to the iris in unit time. *Subset size* measures the difficulty of "breaking" the hash for an adversary that produces random strings. We include both of these numbers in our discussion. We are not aware of algorithmic methods to quickly produce bits distributed according to the iris. There is preliminary work on algorithmically creating synthetic irises [ZS06]. The produced irises are currently distinguishable from real irises [GMF14]. The question of quickly producing outputs of the iris distribution is an important question and will determine the strength of this system going forward. Throughout all of our discussion we assume that the hash function is easy to evaluate (we briefly mentioned memory hard functions in the introduction).

We organize our discussion around versions of the system that improve the system. Any of these versions allow the user to enroll with an arbitrary number of services without affecting security.

3 Basic Implementation

We implement the sample-then-lock construction using the 32768 bit iriscode output. Statistical analysis was done using the ND-0405 dataset [BF16] using the OSIRIS iris recognition [KMSD17] package with transform 5 (which we found to have the best performance on the corpus). This transform is based on a set of filters that produces complex numbers, the transform then ignores the imaginary part and only uses the real component. The ND-0405 dataset includes 356 persons and 64964 total images. Our statistics will include *intraclass* comparisons which are comparisons between two images of the same iris and *interclass* comparisons which are comparisons between two images of different irises. The ND-0405 dataset contains images from both eyes of individuals which are treated as interclass comparisons. We examine the two main obstacles mentioned before:

1. Whether the entropy rate of the iris increases upon subsampling. To determine the security of Gen, we need to show that subsampling yields a high enough entropy rate for the input of the digital locker to be unguessable. In the worst case, sampling only preserves entropy rate. For the human iris which has an entropy rate of 1% producing 40 bits of security would require sampling 4000 bits.

2. For correctness of 50%, how much security can be provided? Importantly we distinguish between *correctness* which is how frequently Rep succeeds and *error rate* which is the distance between w and w'. Decreasing error rate has the effect of increasing effective security (for the same correctness and storage).

Many iris transformations include a *mask* vector that indicates bits that should be ignored due to occlusion. We delay discussing masking until Section 4. The OSIRIS software outputs a binary string for each iris. Figure 2 shows the histograms for fractional Hamming distance between two images of the same individual (*same*) and different individuals (*different*) for the dataset.

This histogram is produced by computing the fractional Hamming distance of every iris with every other iris (for a total of $\approx 10^9$ comparisons). The fractional Hamming distances were then grouped into *interclass* comparisons corresponding to the same iris and *intraclass* comparisons corresponding to different irises.

The error rate of the data is defined as the expected fractional Hamming distance between two images of the same iris. We observed a mean error rate of .32. For different irises, we compute both the interclass mean and interclass variance as $\mu = .494$ and $\sigma = .0008$.

In irises recognition, the standard method for computing the entropy is to compare the interclass histogram with a Binomial distribution with the same mean and variance. If the observed distribution and the Binomial

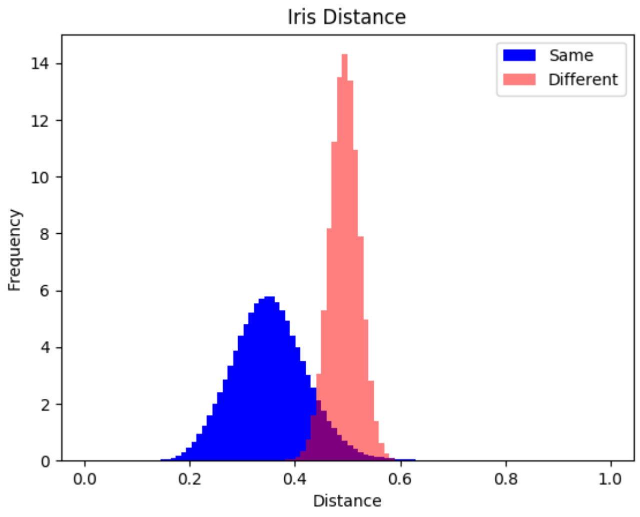


Figure 2: Distribution of distances for the data set.

distribution have very similar histograms, then the observed distribution is assumed to have the same entropy as the Binomial distribution (the technique is described in [Dau04]). This technique is a heuristic. Entropy estimation is a hard problem [VV10, VV11] requiring an exponentially large number of samples in the actual entropy of the distribution.

We performed this heuristic on our setup, generating a binomial distribution with mean $\mu = .494$ and variance $\sigma = .0008$. The statistical distance between the interclass histogram and the binomial distribution was computed with a total statistical distance of .005. The difference between the two probability distribution functions is in the Appendix in Figure 10. From the figure and the total statistical distance the binomial distribution appears a good fit for the observed distribution. Thus, we use the entropy of the Binomial distribution as a stand in for the entropy of the observed distribution. The entropy of the Binomial is calculated using the following equations (where dF stands for degrees of freedom):

$$\begin{aligned}\mathrm{dF} &= \frac{\mu(1-\mu)}{\sigma} = 311\\ \text{entropy} &= (-\mu\log\mu - (1-\mu)\log(1-\mu)) * \mathrm{dF}\\ &= 311.\end{aligned}$$

Subsampling The above analysis establishes a baseline entropy for the OSIRIS transform. To establish the viability of *sample-then-lock* we need to determine if subsets of the OSIRIS transform produce high entropy. As a starting point the OSIRIS output demonstrates an entropy of 311 over a transform of size 32768 for an entropy rate of 1%. In the worst case subsampling only preserves error rate.

The OSIRIS output is produced by convolving a fixed Gabor filter at overlapping regions of the unwrapped iris. Thus, if entropy is geographically distributed throughout the iris, one would expect nearby bits to be correlated. If nearby bits are correlated, subsampling random bits will increase the entropy rate.

To test this hypothesis, we performed the following analysis (for subset size k) with 10 trials for each subset size:

1. Randomly sample k of distinct positions (between 0 and 32767).

2. Compute the intraclass and interclass histograms for the dataset restricted to these positions.

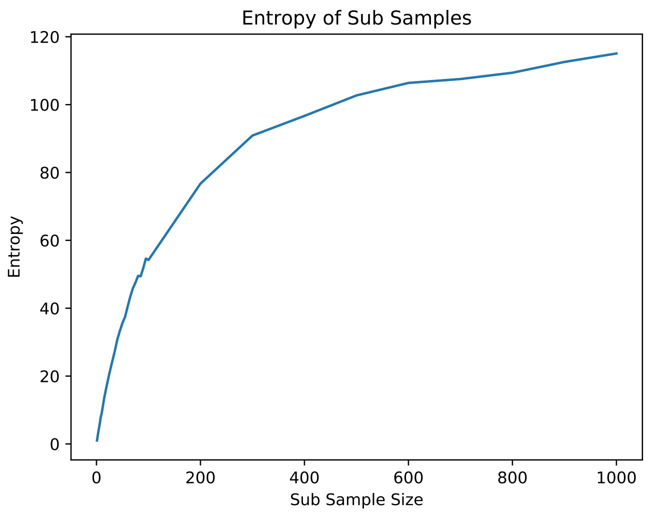


Figure 3: In the worst case subsampling only preserves error rate. For the iris, subsampling greatly increases entropy rate from 1% to over 80%.

3. Compute the μ and σ for the interclass histogram. (Using the same method as in Figure 2).
4. Compute the entropy e_i for trial i.
5. Compute the overall entropy as $e = -\log \mathbb{E}_i\, 2^{-e_i}$

In this last step we average the entropy calculation using average min-entropy [DORS08]. This technique is preferable to averaging the entropies e_i. We are targeting security which requires that the entropy should be high in all settings, not just on average. Consider five possible events where the entropy conditioned on the events is $1, 100, 100, 100, 100$ respectively. Then the average entropy is ≈ 80 while the average min-entropy is ≈ 3.

This analysis was performed for subset sizes $k \in \{1, 2, ..., 10\} \cup \{15, 20, ..., 100\} \cup \{200, 300, ..., 1000\}$ with 10 trials for each size.

Since we are randomly subsampling from a distribution that fits the binomial well, the distribution was assumed to also fit a binomial distribution. These results, shown in Figure 3. We note that the entropy rate is significantly higher than the worst case of 1%. At some points in Figure 3 the entropy rate exceeds 80%.

As mentioned before, the ND-0405 dataset using transformation 5 on OSIRIS has a mean error rate of .32. Using this basic setup we compute parameters as follows:

1. Set target correctness as 50%, number of lockers as 10^6 and mean error rate 32%.
2. Compute maximal subset size k as solution to

$$1 - (1 - (1 - .32)^{10^6})^k = .50.$$

   This yields $k = 32$.
3. Interpolate between $k = 30$ and $k = 35$ on Figure 3 to obtain that entropy/security $e = 26$ when $k = 32$.

The rest of this work is improving the security and subset size parameters.

4 Masking unreliable bits

A technique commonly used to improve iris transforms is called *masking*. (See the work of Bowyer et al. for an overview of iris processing techniques [BHF08].) In most iris transforms in addition to the binary vector w the transform additionally outputs a second vector $mask$. Bits set in $mask$ indicate an error in the transform perhaps due to an eyelash or eyelid (known as an occlusion). Rather than comparing the Hamming distance $d(w, w')$, the authentication only compares locations i where $mask_i = 0 = mask'_i$. The intuition behind the mask vector is that occluded locations are expected to have higher error rates and should be ignored.

A possible way to incorporate $mask$ into *sample-then-lock* is to only sample from positions that are not masked. This technique limits "comparison" to locations where $mask_i = 0$. It is not clear how to incorporate $mask'$.

Instead, we rely on an observation in the iris recognition field, locations to be masked are not uniformly distributed throughout the iris. Rather masked bits usually occur on the top, bottom, inside and outside of the iris [HBF09]. Figure 4 shows the locations that were most frequently masked. In Figure 4, the radius and theta are polar locations with respect to the "center" of the iris. These polar coordinates are mapped to the rectangular space with the y axis representing r and the x axis representing θ. The radius r represents the distance from the pupil and θ represents the angle from $0°$. This figure was calculated by using OSIRIS to compute a mask vector for every image in the ND-0405 dataset. Then, for every location (r, θ) the total number of times that the bit was masked was divided by 64964 (the number of images in the ND-0405 dataset) to normalize the value to between $[0, 1]$. As noted in prior work, the inner and outer rings of the iris are frequently masked in addition to bits with $100 \leq \theta \leq 200$.

We improve *sample-then-lock* by restricting to bits that are unlikely to be masked. Our starting point is a vector pr_{mask} that is a $[0, 1]^{32768}$ vector indicating how frequently each location is masked. We then performed the following analysis for a threshold $thres \in \{1, .0975, .095, .0925, ..., .05, .025\} \cup \{.015\}$. We performed 10 trials for each $thres$:

1. Restrict the input locations to positions j where $pr_{mask,j} > thres$.
2. Compute the mean error rate restricted to these bits.
3. Compute the maximum subset size k.
4. Sample k random bits $\mathcal{I}$ from input locations (where $pr_{mask,j} > thres$) for each trial i.
5. Restrict the input dataset to locations in $\mathcal{I}$. Compute interclass histogram across the entire dataset. Compute $\mu_{thres,i}, \sigma_{thres,i}$ for trial i.
6. Compute the entropy $e_{thres,i}$ for trial i.
7. Compute the overall entropy as $e_{thres} = -\log \mathbb{E}_i \, 2^{-e_{thres,i}}$

The outcome of this analysis is in Table 2. The goal of using masking information was to determine if there were global locations that have higher error rate. Rather than using masking as a proxy for error rate we can directly ask if there are global locations that have higher error rate. There is a strong correlation between these locations, Figure 5 shows the error rate of locations in the same polar coordinates as Figure 4. Thus, we performed similar analysis on bits that have high error rates. This was performed with similar methodology as masking with the following changes:

1. We computed a vector $err\_rate \in [0, 1]^{32768}$.
2. We consider a threshold $thres \in \{1, .99, .98, .97, ..., 0\}$.

Both systems demonstrate an increased subsample size as more bits are filtered. This makes sense, high error rate bits are being removed. However, in both cases we see the entropy of the system increase and then decrease. Our hypothesis is that this is due to having a small area to choose bits from increasing the probability of choosing nearby and correlated bits. This optimum point appears to occur at the same number if we consider bits with a high probability of being masked or a higher error rate.

Pr of mask	Number of Bits	Subsample Size	Entropy
1	32768	32	28
0.9	31810	33	29
0.8	31256	33	29
0.7	30528	33	29
0.6	29455	34	29
0.5	27910	34	30
0.4	26115	35	30
0.3	23861	37	32
0.2	20109	39	31
0.1	15953	41	33
0.075	14572	42	32
0.05	12718	43	32
0.025	9661	44	30
0.015	7619	45	30

Table 2: Security of *sample-then-lock* when restricting to bits that are unlikely to be masked. This table contains a subset of the collected data. The full table is in the Appendix in Table 7.

Error Rate Threshold	Number of Bits	Subsample Size	Entropy
1	32768	32	29
0.475	32635	32	29
0.45	30610	33	29
0.425	26518	35	30
0.4	23417	37	29
0.375	19742	39	32
0.35	16594	41	32
0.325	13723	43	33
0.3	10659	45	31
0.275	6684	47	29
0.25	1876	50	20

Table 3: Security of *sample-then-lock* when restricting to bits that exhibit lower error rate across the data corpus.

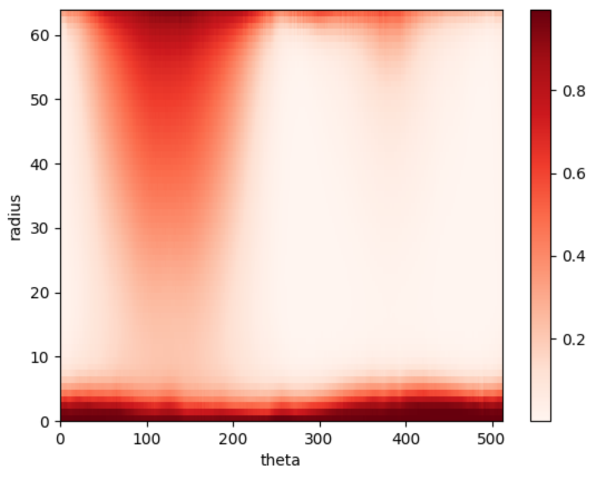


Figure 4: Heatmap of polar coordinate position in the iris (r, θ). The intensity of the pixel indicate the probability that the bit will be "masked" by the transform.

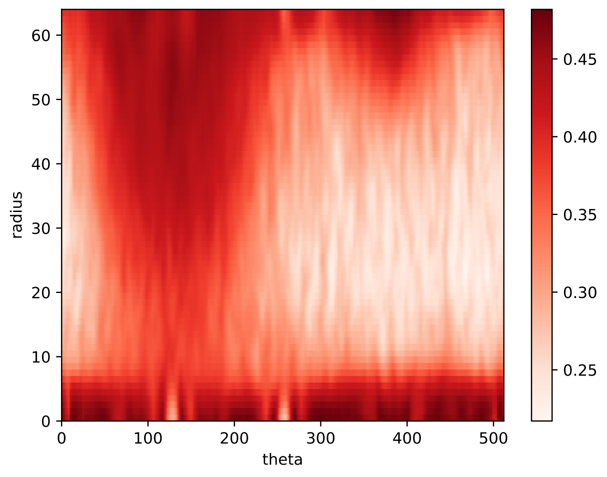


Figure 5: Heatmap of polar coordinate position in the iris (r, θ). The intensity of the pixel indicate the probability that the bit will have an error in a subsequent reading. Note that the scale is different in Figure 4 and this figure.

Our parameters We choose to include the 12000 bits that are least likely to be masked. This was the size that allowed the highest subset size where entropy was close to the maximum. From this point forward, we assume a new iris processing transform that yields a 12000 output corresponding to all bits that are masked at most 5% of the time. For this choice the corresponding subset size is 43.

Note: The reader may notice that the top entries in Tables 2 and 3 do not agree despite them corresponding to the same experiment: subsampling from all 32768 bits of the transform. This difference is due to noise introduced by choosing random subsets of size 32. The results numbers were within .3, the difference of 1 is due to rounding.

5 Strengthening Security using Confidence

The OSIRIS transform is formed by convolution of a 2D Gabor filter with the iris image at a number of positions. This convolution results in an array of complex numbers $a_i + b_i\mathbf{i}$, one for each location where the convolution was centered (the convolution is repeated at a number of positions). Then $a_i + b_i\mathbf{i}$ is converted to two bits, $\texttt{sign}(a_i)$ and $\texttt{sign}(b_i)$. The fractional Hamming distance is then computed across these bits.

We are using OSIRIS transform 5 which only uses the sign of the real component a_i and discards the imaginary component b_i. We found this transform has the best performance among OSIRIS default transforms.

We observe that the magnitude of each coefficient is correlated to error rate of this location in subsequent readings. That is, when $|a_i|$ is large a newly capture image of that iris is less likely to have an error in location i.

Side information which is correlated to w is called *confidence information* and is frequently observed in physical unclonable functions [HRvD+17]. A high confidence value represents a bit with a lower probability of error on subsequent readings. Thus, confidence information can guide subset selection in Gen. However, confidence information may be correlated to w, thus revealing subsets that are biased by confidence information could leak about w.

We are not aware of any iris authentication that uses confidence information. Plaintext systems that compare w and w' achieve good false accept and false reject rates without such information. Most key derivation systems are based on error correcting codes when the input is binary (rather than real-valued).

We incorporate *confidence information* into the sample-then-lock construction. Our analysis only refers to a_i because we are using a transform where b_i is discarded. We leave combination of a_i and b_i to future work.

In order to use $|a_i|$ in our system, we must show the following properties:

A) There exists a negative correlation between $|a_i|$ and error rate in $\texttt{sign}(a_i)$ in subsequent readings.

B) It is possible to restrict to high confidence bits without negatively impacting correctness of *sample-then-lock*.

C) That $|a_i|$ is not correlated with $\texttt{sign}(a_i)$. Furthermore, it may be possible for $|a_i|$ to be correlated with $\texttt{sign}(a_j)$ for $i \neq j$. Any system that uses $|a_i|$ should maintain unpredictability of the values of $\texttt{sign}(a_j)$.

The next three subsections are dedicated to these properties. They correspond to the utility, correctness, and security of confidence information respectively.

5.1 Utility

We begin by showing that $|a_i|$ is correlated with the error rate of $\texttt{sign}(a_i)$. That is, there exists some range of values $[ca, cb]$ such that

$$\begin{aligned} &\Pr[\texttt{sign}(a_i)' \neq \texttt{sign}(a_i) \,|\, |a_i| \in [ca, cb]] \\ <\, &\Pr[\texttt{sign}(a_i)' \neq \texttt{sign}(a_i)]. \end{aligned}$$

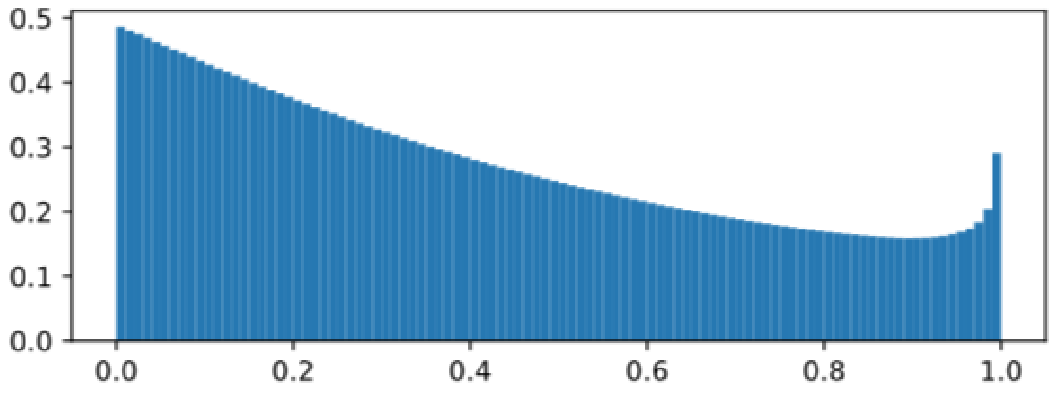
We first compute a frequency histogram of magnitude across the entire data corpus. We derive 100 buckets that represent the smallest 1% of magnitudes, $1\% - 2\%$, etc. To do this we use the following procedure:

1. Input an individual image from the data set.
2. Compute the vectors $\texttt{sign}(a_i)$ and $|a_i|$.
3. Restrict to bits that are unlikely to be masked (where probability of masking is less than 5%).
4. For each bit i
   (a) Compute the bucket of $|a_i|$.
   (b) Compute the error rate of this bit with respect to other images of this eye.
   (c) Add this error rate to the bucket for i.
5. Average across images of this eye.
6. Average across different eyes.

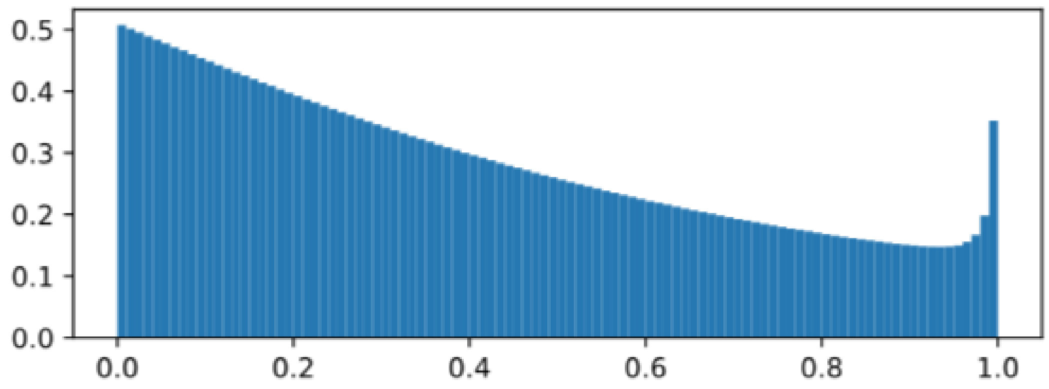
We then compute a histogram of $|a_i|$ vs. the observed error rates $\Pr[\texttt{sign}(a_i)' \neq \texttt{sign}(a_i)]$. We found that this histogram differs based on whether a_i is nonnegative or negative. This determines whether the bit will be mapped to a 1 or 0 respectively. Thus, we present two different histograms. The analysis was recomputed restricting to only nonnegative bits (and then only negative bits).

These histograms are presented in Figure 6. The error rates for low confidence bits are as high as .50. Bits with magnitudes in the .85-.93 bins have average error rate as low as .13. We stress to the reader the difference in the two histograms. In particular, the negative histogram has a sharper “slope” and has a lower error rate in .85-.93 range.

We thus conclude that the confidence value is correlated with the error rate. We now turn to asking two questions, is it possible to incorporate confidence information? what are the security implications of using confidence information?



(a) Nonnegative values: Histogram of confidence vs. error rate.



(b) Negative values: Histogram of confidence vs. error rate.

Figure 6: Histogram of correlation between error rate (y-axis) and magnitude of confidence information. Nonnegative confidence values are on the top histogram and negative confidence values are in the bottom histogram. In each histogram the height of a bar is the number of bits with magnitudes in a range. The x-axis is the percentile of values with a confidence less than a given value.

5.2 Correctness

The lock-then-sample construction needs many bits in the source to create many unique and dissimilar subsets. Recall, that we are computing 10^6 random subsets. For subsets of size k, we expect

$$\Pr[\text{correct}] = 1 - (1 - (1 - \text{error rate})^{10^6})^k.$$

However, if the k approaches the number of bits that are available for subsampling, we expect variance in correctness to increase substantially as selected subsets will overlap. The work of Canetti et al. [CFP+16] (incorrectly) assumes that taking a bit without error does not effect the probability of the next bit having an error. This assumption is not true if the subset size approaches the total number of bits.

After removing bits that are likely to be masked the transform was left with 12000 bits. In this setting, the probability of each pair of subsets sharing a single bit is only 27% (by a birthday bound calculation).

We now determine the minimum number of bits that must be fed to *sample-then-lock* to maintain correctness. Consider selection sizes $sel = 2^{\{6,7,\ldots,13\}}$. To find this minimum number viable sel we do the following:

1. For each eye input an iris code w.

2. Restrict w to a random subset of size sel.

Subsample Size	Correctness
64	17%
128	41%
256	52%
512	56%
1024	60%
2048	60%
4096	60%
8192	60%

Table 4: Relationship between the size of the input to Gen and the correctness of the system. The length of the input w is the first column and the probability that Rep unlocks is second column. The asymptotic behavior of sample-then-lock is not observed if the input to Gen is too small.

3. Run Gen with the restricted w.
4. Execute Rep on all images of the same iris (again restricted to the locations in sel).
5. Record how frequently Rep is successful.
6. Average across images of an iris.
7. Average across irises.

The results are presented in Table 4. We found that $2^{10} = 1024$ bits were needed to achieve target correctness. For this experiment we analyzed how frequently the system was correct for each of the 356 eyes in the ND-0405 dataset. We only performed one trial of this experiment for each selection size.

We used our software implementation (described in Section 7) to perform this analysis rather than observing how frequently an ideal implementation would be correct. The only difference between these two experiments would be due to imperfections in our software or SHA512.

We incorporate confidence into our system as follows. We select a negative and a nonegative positive range that contain the same number of total bits across the corpus. That is, we select equal areas in the two histograms Figure 6. To do this we select a number of bins in the nonnegative histogram that have the minimum error rate and then select a range of negative values with the same total count and minimum integral. This usually means selecting part of a bin for negative values. This approach then yields two ranges $[p_{min}, p_{max}]$ and $[n_{min}, n_{max}]$.

It is necessary to include at least 1024 highly confident bits for the system to maintain correctness. Intuition says taking the minimum size range of confident bits should result in a system with minimum error rate and thus highest security. To verify this intuition, we performed the following analysis for $bins \in \{10, 20, 30, 40, 50, ..., 100\}$.

1. Compute a range $[p_{min,bins}, p_{max,bins}]$ that includes $bins$ with the smallest total integral across the data set.
2. Compute a range $[n_{min,bins}, n_{max,bins}]$ with the same number of entries as $[p_{min,bins}, p_{max,bins}]$ and minimum integral.
3. For each of these range perform 10 trials of:
    (a) For each iris in the data corpus:
        i. Take the first image according to some ordering (we used the first alphabetical file).
        ii. For this file compute the set $\mathcal{I}$ of bits that are unlikely to be masked and where a_i lies within the selected ranges.
        iii. Compute the intra and inter class fractional Hamming distance restricted to $\mathcal{I}$.
    (b) Average across all individuals

# Bits	$[n_{min}, n_{max}]$	$[p_{min}, p_{max}]$	Subset Size	Security
12000	[-21900, -26]	[6,21900]	43	35
10800	[-21900,-36]	[68,21900]	46	37
9600	[-21900, -99]	[131, 21900]	50	39
8400	[-21900, -165]	[198, 21900]	55	41
7200	[-2132,-228]	[263,2132]	60	43
6000	[-2132, -307]	[343, 2132]	65	45
4800	[-2132, -399]	[437, 2132]	70	44
3600	[-2132, -513]	[540, 1726]	75	42
2400	[-1726, -654]	[676,1522]	80	40
1200	[-1522, -879]	[828,1206]	83	35

Table 5: Security and subset sizes based on confidence ranges that include *# bits* in expectation. The positive and negative ranges are the real values that in expectation select that many bits.

(c) Compute the interclass mean μ_{bins} and variance σ_{bins}.

(d) Calculate the degrees of freedom and entropy as above.

(e) Compute the average min-entropy.

The result of this analysis is in Table 5. This table contains the computed ranges as well as the maximum subset size and the resulting entropy. Recall that 10 bins of the histogram corresponds to 1200 bits, 20 bins to 2400 bits and all 100 bins to 12000 bits.

The minimum number of bins produced the lowest effective error rate and thus the highest subsample size k. Using only 1200 bits allows for a subsample size of 83. Surprisingly, the entropy of the input subset is **not** maximized by taking ten bins. We found that including 6000 confident bits produced an input entropy of 45 with a subset size of 65. There are a number of possible causes for this behavior, i) highly confident bits could also be bits that do not vary across the data corpus ii) highly confident bits tend to be in close proximity and thus are redundant iii) when restricting to a smaller set a “bad” area of an image is more likely to be resulting in one trial with low entropy which will drop average min-entropy. We have not identified a root cause.

When we set ranges to select 6000 confident bits on average it is very rare for an image to have less than 1024 bits in this range. Thus, we do not expect correctness to be effected by restricting to bits in this range.

A priori, there is no reason to expect that high confidence bits have a similar probability density function as a binomial distribution. To establish this fact we first compute a histogram for inter/intra class distance. We do this using the following method:

1. Input an individual image from the data set.
2. Restrict the image to locations whose magnitudes lie within $[343, 2132]$ or $[-2132, -307]$ (and are not filtered by masking). Denote this location set by $\mathcal{I}$. Note that $|\mathcal{I}|$ may be larger or smaller than 6000.
3. Compute interclass and intraclass histograms for the entire corpus restricted to $\mathcal{I}$.
4. Average across images from the same iris.
5. Average across irises.

This histogram is in Figure 7. For different irises, the results showed a mean of $\mu = .495$ and variance of $\sigma = .00264$. We compared the interclass distribution with a binomial distribution with mean $\mu = .495$ and variance $\sigma = .00264$ (as in [Dau04]). The total statistical distance is .087. The difference between the two probability distribution functions is in the appendix in Figure 11.

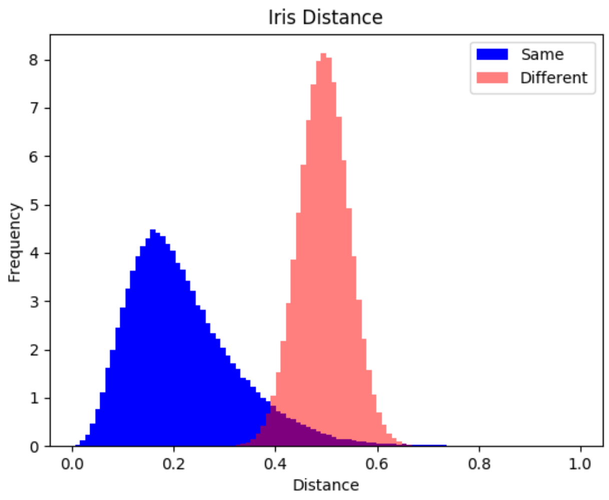


Figure 7: Distribution of distances for the data set restricted to high confidence bits. This figure is formed by selecting all locations of an image that have magnitude between $[343, 2132]$ or $[-2132, -307]$ and then computing the fractional Hamming distance with all other images in the data set restricted to these locations.

5.3 Security

Using confident bits has a subtle but important impact on security. The adversary is able to see which bits are selected in subsets (the locations are considered public). This means that the adversary can infer which bits are high confidence. We ensured that a bit is equally likely to be 1 and 0 based on being high confidence. However, this does not rule out other types of correlations. For example it might be that when bit 0 is high confidence it always takes the same value as bit 1000. We do not expect this behavior, however, it is an additional assumption. Subsets are now being computed based on the value of the iris (instead of randomly). This leads us to the following assumption.

Assumption 1. *Let $\mathcal{I}$ the random variable of the location of high confidence bits for an iris. All information about $W_{\mathcal{I}} \overset{def}{=} \mathtt{sign}(\vec{a})_{\mathcal{I}}$ is contained in the values of other irises restricted to the locations in $\mathcal{I}$.*

Future research may show that there are other ways to utilize the location of the high confidence bits despite the fact that they vary widely across individuals readings. This assumption also implies that it is safe to reuse confidence information. If all information about the value $\mathtt{sign}(\vec{a})_I$ is contained in the value of other irises this immediately implies that having multiple high confidence sets is not helpful. Assumption 1 allows us to state the security of our system[7]:

Theorem 1. *If Assumption 1 holds and the average min-entropy of high confidence bits is 45, denoted $\tilde{H}_\infty(W_{\mathcal{I}}|\mathcal{I}) \geq 45$, and HMAC-SHA-512 serves as a composable digital locker then the construction is a reusable fuzzy extractor requiring 2^{45} queries to the underlying hash function to break security.*

We believe that 45 is a lower bound for effective security. The security of the digital locker is defined by the number of queries that must be made to the hash function. We believe a more accurate time bound is the time to sample 65 bit strings that are distributed according to the iris. There are tools to quickly generate guesses according to the password distribution but we are not aware of comparable tools for the iris.

[7]This theorem relies on the formal definitions of a reusable fuzzy extractor and average min-entropy [DORS08]. These definitions are deferred to the Appendix A for space considerations.

Unlinkability Reusability and unlinkability are two different properties. Unlinkability prevents an adversary from telling if two enrollments correspond to the same physical source [CS08, KBK+11]. Our construction without confidence information satisfies unlinkability. More study is necessary to determine if an adversary can break linkability when confidence information is used. When confidence information is used an adversary may be able to link by checking for the size of overlap between sampled positions.

Weakening the assumption Using recent cryptographic work it is be possible to weaken Assumption 1. The recent work of Wichs and Zerdelis [WZ17] shows how to obfuscate "compute-and-compare" programs under the LWE assumption [Reg09]. This work uses techniques from generic obfuscation but is based on a standard assumption. A compute and compare program is one in which the program has three stored values f, y, z. The value f is interpreted as function or program while y and z are fixed bit strings. The obfuscated program takes input x, computes $f(x)$, and checks whether $f(x) = y$, and outputs z if they are equal, otherwise nothing is output. Their construction is secure (additionally assuming the existence of extremely lossy functions [Zha16]) as long as y has pseudoentropy conditioned on f. Pseudoentropy is the computational version of entropy and is defined in [HILL99]. We can adapt our construction into a compute-and-compare program. We first describe how to construct f:

1. Input a_i, w_i.
2. Generate two keys $\mathsf{Key}_1, \mathsf{Key}_2$.
3. For $i = 1, ..., \ell$:
   (i) Choose $1 \leq j_{i,1}, ..., j_{i,k} \leq |w|$ where $a_{j_i} \in [pa, pb]$ or $a_{j_{i_k}} \in [na, nb]$.
   (ii) Choose a hash key h_i.
   (iii) Create multiplexor m_i from $|w|$ to k bits with selections bits of $j_{i,1}, ..., j_{i,k}$
   (iv) Set $c_i = \mathtt{Hash}(h_i, m_i(w))$.
   (v) Set $p_i = (0^{256}||\mathsf{Key}) \oplus c_i$.

The function f then does the following

1. Takes as input w'.
2. Runs w' through the various multiplexors to get the selected subsets.
3. Runs the resulting hashes
4. If any hash h_i has that property that $h_i \oplus c_i$ begins with 0^{128} output Key'

We then define $y = \mathsf{Key}_1$ and $z = \mathsf{Key}_2$. This approach can be used to hide which bits are being selected as long as the resulting Key_1 has high entropy conditioned on the location of those bits. This allows us to weaken our assumption to: $W_{\mathcal{I}}|\mathcal{I}$ has high pseudoentropy. While this potential change has security benefits there are several drawbacks:

1. The approach of Wichs and Zerdelis uses general branching programs that have been used in constructions of obfuscation from multilinear maps. This increases the size of the obfuscation to beyond practical terms.
2. Wichs and Zerdelis do not make any argument that their obfuscation is reusable. Thus, this adapted approach would only be helpful for authentication with a single provider.

Given these two drawbacks we do not recommend using compute-and-compare obfuscation. However, this is a promising research direction as more classes of function have obfuscation based on standard assumptions.

6 Adding a password

Many multi factor authentication systems do not achieve "additive security." Consider a strawman authentication technique: 1) a user inputs a password pw and 2) an iris w. Currently, the password would be compared to salted hash and the iris is compared in plaintext. One of these comparisons has to be done first. In either case based on time or error messages it is often possible to perform a brute force search on each factor separately.

In theory, password can be considered a noiseless input with entropy to any fuzzy extractor and strengthen key derivation. However, previous fuzzy extractors separate the error-correction from key derivation process using two distinct primitives called secure sketch [DORS08] and randomness extractor respectively [NZ93]. The secure sketch is responsible for mapping w' back to the input w, while the randomness extractor converts entropy into a random key. In such a process, In a multi factor scheme password can only be incorporated into the randomness extractor (and not used in the secure sketch). Many secure sketches are called well-formed meaning they report an error when $d(w, w') > t$. If such an error is visible to the adversary they can separate searching w' and searching for the password. Hiding such timing channels is notoriously difficult.

Our construction does not suffer from this problem. "Error-correction" and key derivation are performed simultaneously. The password can be prepended as input to each hash invocation without affecting storage or computational requirements. Recent estimates place password entropy at 34 bits [KSK+11], giving the hybrid system 79 bits of entropy.

7 Implementation

We implemented our construction in a Python library. Previous implementations of fuzzy extractors required expertise in error-correcting codes, forcing the user to understand finite fields. Our construction is much simpler, requiring only repeated evaluation of a hash function. As evidence of this, the entire key derivation system requires less than 100 lines of python code with dependencies on numPy (for array manipulation), random, and hashlib. A basic version of Gen is below. We do not include filtering for masked or confident bits in our presented code for simplicity.

```
def gen(self, bits, locker_size, lockers):
  length = self.hash().digest_size
  r = self.generate_sample(size=length/2)
  zeros = bytearray([0 for x in range(length/2)])
  check = zeros + r
  seeds = self.generate_sample(length=lockers)
  vs = numPy.array([random.sample(pick_range, locker_size) \
                for x in range(lockers)])
  p = []
  for x in range(lockers):
    v_i = numPy.array([bits[y] for y in vs[x]])
    seed = seeds[x]
    h = bytearray(hmac.new(seed, v_i, self.hash).digest())
    c_i = self.xor(check , h)
    p.append((c_i, positions[x], seed))
  return r, p
```

This version does not consider confidence information. Our Gen code is single threaded because the majority of execution time is spent in system calls to achieve the randomness needed for the subsets $j_{i,1}, ..., j_{i,k}$.

We implemented two versions of Rep. The first is single threaded and very simple.

```
def rep(self, bits, p):
  count = 0
  for c_i, positions, seed in p:
   v_i = numPy.array([bits[x] for x in positions])
   h = bytearray(hmac.new(seed, v_i, self.hash).digest())
   res = self.xor(c_i, h)
   if(self.check_result(res)):
       return res[len(res)/2:]
   count += 1
  return None
```

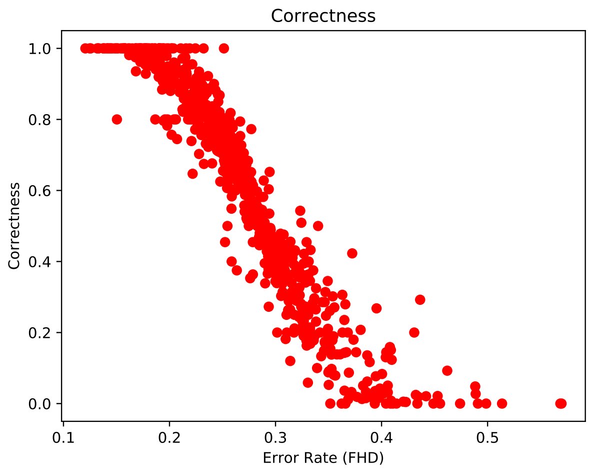


Figure 8: Correlation between correctness of Rep and the error rate of an individual's eye. The mean error rate across the corpus is 60%. Based on subset size of 43 bits with the 12000 bits that have probability of being masked of at less than 5%.

The Rep functionality is embarrassingly parallel. The code launches a number of threads which divide the hashed value. Rep succeeds when one of these threads returns. This version is presented in the appendix in Figure 12.

This code was not intended for high efficiency applications. We expect an optimized low level implementation to be orders of magnitude fast than our current implementation. We evaluated two versions of our implementation 1) one that filtered bits that were likely to be masked and 2) one that additionally incorporated confidence information.

Our implementation was tested with these parameters: 1) starting with 12000 bits that are unlikely to be masked 2) using a subset size of 43 bits. Our target correctness was 50% across the corpus. Our mean correctness was actually, 60%. As expected the correctness is highly correlated with the error rate of the underlying iris. This correlation is demonstrated in Figure 8. We also computed performance results with Gen taking on average 220s and Rep taking on average 12s. The time for Rep is an upper bound as the time was derived from an average of multiple irises that did not authenticate. Specifically, as each of the irises did not authenticate, Rep had to go through all 10^6 lockers. Practically, the average time for Rep should be 50% faster as one expects an opening locker to be find in half the time. This performance analysis was performed on a Dell Precision Tower 7000 Series with a Xeon E5-2620 v4 processor and 64GB of RAM.

Confidence Evaluation For evaluating confidence, we tested the system with a confidence range expected to yield 2000 bits per iris. This is because this was the minimum size range where 90% of images had at least 1024 bits (where correctness of the scheme begins to degrade). We tested this range instead of 6000 confident bits because fewer bits are likely to hurt correctness. If correctness is preserved for 2000 bits it will also be preserved for 6000 bits.

Our python implementation was tested with these parameters: 1) starting with 12000 bits that are unlikely to be masked 2) filter all bits whose confidence is outside both $[pa, pb] = [676.25, 1386.75]$ and $[na, nb] = [-1726.00, -673.12]$. and 2) using a subset size of 81 bits. Our mean correctness was 59%. As expected the correctness is highly correlated with the error rate of the underlying iris. This correlation is demonstrated in Figure 9. Adding confidence information does not significantly impact time to run Gen or Rep.

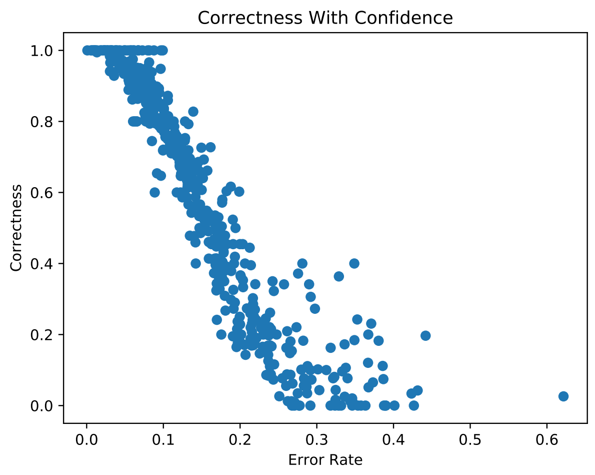


Figure 9: Correlation between correctness of Rep and the error rate of an individual's eye. The mean error rate across the corpus is 60%. Based on subset size of 81. Subsets selected by first 1) removing bits that are likely to be masked and 2) only selecting from bits in high confidence range.

Scheme	Security	Subset Size
Basic	26	32
Masking	32	43
Confidence (recommended)	45	65
Confidence (opt. for subset size)	35	83

Table 6: Summary of the security of the different systems described in this work. We recommend the use of the confidence system in the third row. The fourth row is a system that maximizes subset size.

8 Conclusion

In this work we have described the first key derivation system for the human iris that allows an individual to enroll with multiple devices and services. Our system is built from the recent *sample-then-lock* construction of Canetti et al. [CFP+16]. However, for meaningful security we needed to combine the image processing with the cryptographic techniques. These combinations produce several different versions of the systems (summarized in Table 6). Our security is based on the error rates produced by the open source OSIRIS iris processing library. We expect high security for a transform with lower error rates.

We believe that biometric authentication is inevitable, with deployment on most mobile devices. However, recent cryptographic advances put securing biometric authentication in reach. We hope this work encourages further research into the iris but also in deriving keys from fingerprints and facial geometry.

Acknowledgements

We thank Leonid Reyzin and Alexander Russell for helpful discussions and insights.

References

[ABC+16] Quentin Alamélou, Paul-Edmond Berthier, Stéphane Cauchie, Benjamin Fuller, and Philippe Gaborit. Reusable fuzzy extractors for the set difference metric and adaptive fuzzy extractors. 2016.

[ACEK17] Daniel Apon, Chongwon Cho, Karim Eldefrawy, and Jonathan Katz. Efficient, reusable fuzzy extractors from LWE. In *International Conference on Cyber Security Cryptography and Machine Learning*, pages 1–18. Springer, 2017.

[BA12] Marina Blanton and Mehrdad Aliasgari. On the (non-) reusability of fuzzy sketches and extractors and security improvements in the computational setting. *IACR Cryptology ePrint Archive*, 2012:608, 2012.

[BA13] Marina Blanton and Mehrdad Aliasgari. Analysis of reusability of secure sketches and fuzzy extractors. *IEEE transactions on information forensics and security*, 8(9-10):1433–1445, 2013.

[BCC$^+$07] Julien Bringer, Hervé Chabanne, Gérard Cohen, Bruno Kindarji, and Gilles Zémor. Optimal iris fuzzy sketches. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on*, pages 1–6. IEEE, 2007.

[BF16] Kevin W Bowyer and Patrick J Flynn. The ND-IRIS-0405 iris image dataset. *arXiv preprint arXiv:1606.04853*, 2016.

[BH09] Marina Blanton and William MP Hudelson. Biometric-based non-transferable anonymous credentials. In *Information and Communications Security*, pages 165–180. Springer, 2009.

[BHF08] Kevin W Bowyer, Karen Hollingsworth, and Patrick J Flynn. Image understanding for iris biometrics: A survey. *Computer vision and image understanding*, 110(2):281–307, 2008.

[BHF13] Kevin W Bowyer, Karen P Hollingsworth, and Patrick J Flynn. A survey of iris biometrics research: 2008–2010. In *Handbook of iris recognition*, pages 15–54. Springer, 2013.

[BHVOS12] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012.

[Boy04] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 82–91, New York, NY, USA, 2004. ACM.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security (CCS)*, pages 62–73, 1993.

[BS00] Sacha Brostoff and M.Angela Sasse. Are passfaces more usable than passwords?: A field trial investigation. *People and Computers*, pages 405–424, 2000.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *Advances in Cryptology–EUROCRYPT 2008*, pages 489–508. Springer, 2008.

[CFP$^+$16] Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam Smith. Reusable fuzzy extractors for low-entropy distributions. In *Advances in Cryptology–Eurocrypt 2016*, pages 117–146. Springer, 2016.

[CKVW10] Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In *Theory of Cryptography (TCC)*, pages 52–71, 2010.

[CS08] F Carter and A Stoianov. Implications of biometric encryption on wide spread use of biometrics. In *EBF Biometric Encryption Seminar (June 2008)*, 2008.

[Dak09] Ramzi Ronny Dakdouk. *Theory and Application of Extractable Functions*. PhD thesis, Yale University, 2009. `http://www.cs.yale.edu/homes/jf/Ronny-thesis.pdf`.

[Dau04] John Daugman. How iris recognition works. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):21 – 30, January 2004.

[DGV$^+$16] Jeroen Delvaux, Dawu Gu, Ingrid Verbauwhede, Matthias Hiller, and Meng-Day Mandel Yu. Efficient fuzzy extraction of PUF-induced secrets: Theory and applications. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pages 412–431. Springer, 2016.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DRS] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology –Eurocrypt*.

[FMR13] Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. Computational fuzzy extractors. In *Advances in Cryptology-ASIACRYPT 2013*, pages 174–193. Springer, 2013.

[GKTF16] Zimu Guo, Nima Karimian, Mark M Tehranipoor, and Domenic Forte. Hardware security meets biometrics for the age of IoT. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pages 1318–1321. IEEE, 2016.

[GM84] Alexander Grossmann and Jean Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM journal on mathematical analysis*, 15(4):723–736, 1984.

[GMF14] Javier Galbally, Sébastien Marcel, and Julian Fierrez. Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition. *IEEE transactions on image processing*, 23(2):710–724, 2014.

[HAD06] Feng Hao, Ross Anderson, and John Daugman. Combining crypto with biometrics effectively. *IEEE Transactions on Computers*, 55(9):1081–1088, 2006.

[HBF09] Karen P Hollingsworth, Kevin W Bowyer, and Patrick J Flynn. The best bits in an iris code. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):964–973, 2009.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HRvD$^+$17] Charles Herder, Ling Ren, Marten van Dijk, Meng-Day Mandel Yu, and Srinivas Devadas. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Transactions on Dependable and Secure Computing*, 14(1):65–82, 2017.

[ICF$^+$15] Gene Itkis, Venkat Chandar, Benjamin W Fuller, Joseph P Campbell, and Robert K Cunningham. Iris biometric security challenges and possible solutions: For your eyes only? using the iris as a key. *IEEE Signal Processing Magazine*, 32(5):42–53, 2015.

[Jos15] Simon Josefsson. The memory-hard argon2 password hash function. *memory*, 2015.

[KBK$^+$11] Emile JC Kelkboom, Jeroen Breebaart, Tom AM Kevenaar, Ileana Buhan, and Raymond NJ Veldhuis. Preventing the decodability attack based cross-matching in a fuzzy commitment scheme. *Information Forensics and Security, IEEE Transactions on*, 6(1):107–121, 2011.

[KCK$^+$08] Sanjay Kanade, Danielle Camara, Emine Krichen, Dijana Petrovska-Delacrétaz, and Bernadette Dorizzi. Three factor scheme for biometric-based cryptographic key regeneration using iris. In *Biometrics Symposium, 2008. BSYM'08*, pages 59–64. IEEE, 2008.

[KMSD17] Emine Krichen, Anouar Mellakh, Sonia Salicetti, and Bernadette Dorizzi. Osiris (open source for iris) reference system, 2017.

[KSK$^+$11] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2595–2604. ACM, 2011.

[LPS04] Benjamin Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In *Advances in Cryptology–EUROCRYPT 2004*, pages 20–39. Springer, 2004.

[NZ93] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, pages 43–52, 1993.

[PJ16] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. Technical report, 2016.

[PJA10] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[PPJ03] Salil Prabhakar, Sharath Pankanti, and Anil K Jain. Biometric recognition: Security and privacy concerns. *IEEE Security & Privacy*, 1(2):33–42, 2003.

[PRC15] Vishal M Patel, Nalini K Ratha, and Rama Chellappa. Cancelable biometrics: A review. *IEEE Signal Processing Magazine*, 32(5):54–65, 2015.

[PSJO$^+$06] P. Jonathan Phillips, W. Todd Scruggs, Alice J. O'Toole, Patrick J. Flynn, Kevin W. Bowyer, Cathy L. Schott, and Matthew Sharpe. FRVT 2006 and ICE 2006 large-scale experimental results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[PUB95] FIPS PUB. Secure hash standard. *Public Law*, 100:235, 1995.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[sam] Samsung pass. Internet website; accessed 21 September 2017.

[STP09] Koen Simoens, Pim Tuyls, and Bart Preneel. Privacy weaknesses in biometric sketches. In *IEEE Symposium on Security and Privacy*, pages 188–203. IEEE, 2009.

[Vad03] Salil P Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In *Advances in Cryptology-CRYPTO 2003*, pages 61–77. Springer, 2003.

[VV10] Gregory Valiant and Paul Valiant. A CLT and tight lower bounds for estimating entropy. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 17, page 9, 2010.

[VV11] Gregory Valiant and Paul Valiant. Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 685–694. ACM, 2011.

[WJMM05] James Wayman, Anil Jain, Davide Maltoni, and Dario Maio. An introduction to biometric authentication systems. *Biometric Systems*, pages 1–20, 2005.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. *IACR Cryptology ePrint Archive*, 2017:276, 2017.

[Zha16] Mark Zhandry. The magic of elfs. In *Annual Cryptology Conference*, pages 479–508. Springer, 2016.

[ZS06] Jinyu Zuo and Natalia A Schmid. A model based, anatomy based method for synthesizing iris images. In *International Conference on Biometrics*, pages 428–435. Springer, 2006.

A Cryptographic Definitions and Tools

We generally use capital letters to refer to random variables. For a set of indices J, X_J is the restriction of X to the indices in J. U_n denotes the uniformly distributed random variable on $\{0,1\}^n$. Unless otherwise noted logarithms are base 2.

Definition 1. *The* min-entropy *of X is* $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X = x])$*,*

Definition 2. *[DORS08, Section 2.4] The* average (conditional) *min-entropy of X given Y is* $\tilde{\mathrm{H}}_\infty(X|Y) = -\log(\mathbb{E}_{y\in Y} \max_x \Pr[X = x|Y = y])$*.*

The *statistical distance* between random variables X and Y with the same domain is $\Delta(X,Y) = \frac{1}{2}\sum_x |\Pr[X = x] - \Pr[Y = x]|$.

For a distinguisher D we write the *computational distance* between X and Y as $\delta^D(X,Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|$ (we extend it to a class of distinguishers $\mathcal{D}$ by taking the maximum over all distinguishers $D \in \mathcal{D}$). We denote by $\mathcal{D}_s$ the class of randomized circuits which output a single bit and have size at most s.

Fuzzy Extractors In this section we define computational fuzzy extractors [FMR13]. Similar definitions for information-theoretic fuzzy extractors can be found in the work of Dodis et al. [DORS08, Sections 2.5–4.1]. The definition of computational fuzzy extractors allows for a small probability of error.

Definition 3. *[FMR13, Definition 4] Let $\mathcal{W}$ be a family of probability distributions over metric space* $(\mathcal{M}, \mathsf{dis})$*. A pair of randomized procedures "generate" (*Gen*) and "reproduce" (*Rep*) is an* $(\mathcal{M}, \mathcal{W}, \kappa, t)$-computational fuzzy extractor *that is* $(\epsilon_{sec}, s_{sec})$*-hard with error δ if* Gen *and* Rep *satisfy the following properties:*

- *The generate procedure* Gen *on input $w \in \mathcal{M}$ outputs an extracted string $r \in \{0,1\}^\kappa$ and a helper string $p \in \{0,1\}^*$.*
- *The reproduction procedure* Rep *takes an element $w' \in \mathcal{M}$ and a bit string $p \in \{0,1\}^*$ as inputs. The* correctness *property guarantees that if* $\mathsf{dis}(w,w') \le t$ *and* $(r,p) \leftarrow \mathsf{Gen}(w)$*, then* $\Pr[\mathsf{Rep}(w',p) = r] \ge 1-\delta$*, where the probability is over the randomness of* $(\mathsf{Gen}, \mathsf{Rep})$*.*
- *The* security *property guarantees that for any distribution $W \in \mathcal{W}$, the string r is pseudorandom conditioned on p, that is* $\delta^{\mathcal{D}_{s_{sec}}}((R,P),(U_\kappa,P)) \le \epsilon_{sec}$*.*

Reusable Fuzzy Extractors A desirable feature of fuzzy extractors is reusability [Boy04, CFP+16]. Intuitively, it is the ability to support multiple enrollments of the same iris.

The algorithm Gen may be run multiple times on correlated readings $w_1, ..., w_\rho$ of a given source. Each time, Gen will produce a different pair of values $(r_1,p_1), ..., (r_\rho, p_\rho)$. Security for each extracted string r_i should hold even in the presence of all the helper strings $p_1, \ldots, p_\rho$ and all keys r_j such that $j \neq i$.

Definition 4 (Reusable Fuzzy Extractors)**.** *Let $\mathcal{W}$ be a family of distributions over $\mathcal{M}$. Let* $(\mathsf{Gen}, \mathsf{Rep})$ *be a* $(\mathcal{M}, \mathcal{W}, \kappa, t)$*-computational fuzzy extractor that is* $(\epsilon_{sec}, s_{sec})$*-hard with error δ. Let* $(W^1, W^2, \ldots, W^\rho)$ *be ρ correlated random variables such that each $W^j \in \mathcal{W}$. Let D be an adversary. Define the following game for all $j = 1, ..., \rho$:*

- ***Sampling*** *The challenger samples $w^j \leftarrow W^j$ and $u \leftarrow \{0,1\}^\kappa$.*
- ***Generation*** *The challenger computes* $(r^j, p^j) \leftarrow \mathsf{Gen}(w^j)$*.*
- ***Distinguishing*** *The advantage of D is*

$$\begin{aligned} Adv(D) &\overset{def}{=} \\ &\Pr[D(r^1, ..., r^{j-1}, r^j, r^{j+1}, ..., r^\rho, p^1, ..., p^\rho) = 1] \\ &- \Pr[D(r^1, ..., r^{j-1}, u, r^{j+1}, ..., r^\rho, p^1, ..., p^\rho) = 1]. \end{aligned}$$

$(\mathsf{Gen}, \mathsf{Rep})$ *is* $(\rho, \epsilon_{sec}, s_{sec})$*-reusable if for all* $D \in \mathcal{D}_{s_{sec}}$ *and for all* $j = 1, ..., \rho$*, the advantage is at most* ϵ_{sec}*.*

Cryptographic Tools Our construction uses digital lockers, which are computationally secure symmetric encryption schemes that retain security even when used multiple times with correlated and weak (i.e., nonuniform) keys [CKVW10]. Digital lockers have two important properties:

1. Obtaining any information about the "locked" value is only possible by guessing the key.

2. It is possible to detect the wrong key with high probability.

We use notation $\mathsf{c} = \mathsf{lock}(\mathsf{key}, \mathsf{val})$ for the algorithm that performs the locking of the value val using the key key, and $\mathsf{unlock}(\mathsf{key}, \mathsf{c})$ for the algorithm that performs the unlocking (which will output val if key is correct and $\perp$ with high probability otherwise).

Digital lockers can be constructed from variants of the Diffie-Hellman assumption [CD08]. Our construction assumes that hash functions can be used to construct digital lockers. Let H be a cryptographic hash function. The locking algorithm $\mathsf{lock}(\mathsf{key}, \mathsf{val})$ outputs the pair $\mathsf{nonce}, H(\mathsf{nonce}, \mathsf{key}) \oplus (0^{\mathsf{s}}||\mathsf{val})$, where nonce is a nonce, $||$ denotes concatenation, and s is a security parameter.

As long as the entropy of key is superlogarithmic, the adversary has negligible probability of finding the correct key; and if the adversary doesn't find the correct key, then the adversarial knowledge about key and val is not significantly affected by this locker. Concatenation with 0^{s} is used to make sure that unlock can tell (with certainty $1 - 2^{-\mathsf{s}}$) when the correct value is unlocked. This construction was proposed in [BR93] and shown to be secure in the random oracle model by Lynn, Prabhakaran, and Sahai [LPS04, Section 4] It is plausible that in the standard model (without random oracles) hash functions provide the necessary security [CD08, Section 3.2], [Dak09, Section 8.2.3].

B Additional Statistical analysis

In this section we provide additional statistical analysis. First, our analysis of the security of masking in Table 2 was abbreviated so we provide the full table in Table 7. Second, in the body of this paper we computed the statistical distance between the observed interclass distribution and the binomial distribution in two settings:

1. When all bits are included in the comparison as described in Section 3 (interclass distribution from Figure 2).

2. When restricted to highly confidence bits of an image as described in Section 5.

We provide a full plot of these statistical distances in Figures 10 and 11 respectively.

C Source Code

In Section 7, we code for our single threaded Gen and Rep functionality. To show the embarrassingly parallel nature of Rep the parallel version of our implementation is in Figure 12.

Masking

Pr of mask	Number of Bits	Subsample Size	Entropy
1	32768	32	28
0.975	32341	33	30
0.95	32109	33	30
0.925	31887	33	30
0.9	31810	33	29
0.875	31695	33	30
0.85	31542	33	29
0.825	31420	33	30
0.8	31256	33	29
0.775	31099	33	29
0.75	30913	33	29
0.725	30725	33	30
0.7	30528	33	29
0.675	30283	33	29
0.65	30008	34	30
0.625	29736	34	30
0.6	29455	34	29
0.575	29113	34	30
0.55	28745	34	30
0.525	28340	34	30
0.5	27910	34	30
0.475	27483	35	30
0.45	27039	35	30
0.425	26618	35	31
0.4	26115	35	30
0.375	25618	36	31
0.35	25097	36	30
0.325	24462	36	31
0.3	23861	37	32
0.275	23196	37	30
0.25	22381	37	31
0.225	21353	38	32
0.2	20109	39	31
0.175	19144	39	32
0.15	18139	40	32
0.125	17089	40	32
0.1	15953	41	33
0.075	14572	42	32
0.05	12718	43	32
0.025	9661	44	30
0.015	7619	45	30

Table 7: Full table of results for restricting to bits that are likely to be masked. Expands on the data in Security of *sample-then-lock* when restricting to bits that are unlikely to be masked.

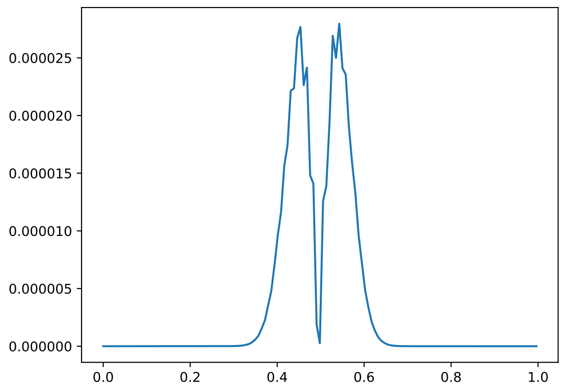


Figure 10: The statistical distance between the interclass comparisons of the base data set and a binomial distribution with the same mean and variance. The x-axis is the Hamming distance (resp. the normalized value of the binomial distribution). The y-axis is the statistical distance between the two distributions.

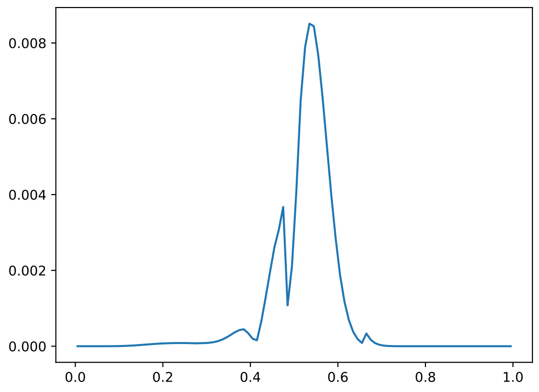


Figure 11: The statistical distance between the interclass comparisons restricted to “confident” bits compared to a binomial distribution with the same mean and variance. The x-axis is the Hamming distance (resp. the normalized value of the binomial distribution). The y-axis is the statistical distance between the two distributions.

```
def rep(self, bits, p, num_processes=1):
    finished = multiprocessing.Array('b', False)
    split = np.array_split(p, num_processes)
    finished = multiprocessing.Manager().list(\
                    [None for x in range(num_processes)])
    processes = []
    for x in range(num_processes):
        p = multiprocessing.Process(\
         target=self.rep_proc, args=(bits, split[x], fin, x))
        processes.append(p)
        p.start()
    for p in processes:
        p.join()
    if(any(fin)):
       return next(item for item in finished\
          if item is not None)
    return None

def rep_proc(self, bits, p, finished, process_id):
    counter = 0
    for c_i, positions, seed in p:
        v_i = np.array([bits[x] for x in positions])
        h = bytearray(hmac.new(seed, v_i, self.hash).digest())
        res = self.xor(c_i, h)
        if(self.check_result(res)):
            fin[process_id] = res[len(res)/2:]
            return
        counter += 1
        if(counter == 1000):
            if(not any(finished)):
                counter = 0
            else:
                return
```

Figure 12: Code for multi-threaded Rep functionality. Some simplifications have been made for space considerations.