Reusable Authentication from the Iris

Sailesh Simhadri, James Steel, and Benjamin Fuller

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

May 14, 2018

Abstract

Mobile platforms use biometrics for authentication. Biometrics exhibit noise between repeated readings. Due to the noise, biometrics are stored in plaintext increasing risk if a device is compromised. Since biometrics cannot be regenerated or refreshed, they will be reused, increasing the impact of such a compromise. Fuzzy extractors derive a stable cryptographic key from biometrics (Dodis et al., Eurocrypt 2004). Previous works claim biometric key derivation systems using fuzzy extractors but these works either assume an adversary model where plaintext biometric storage is secure or have incorrect analysis. In addition, no construction handles the case of biometric reuse.

The goal of this work is to derive keys from an actual biometric with formal and explicit conditions for security. We focus on the iris due to its strong uniqueness (Prabhakar, Pankanti, and Jain, IEEE S&P 2003). We build an iris key derivation system with 45 bits of security even when the iris is reused. Our starting point is *sample-then-lock*, a recent fuzzy extractor due to Canetti et al. (Eurocrypt 2016). Achieving satisfactory parameters requires modifying and coupling the image processing and cryptographic algorithms. Our system is based on repeated hashing which simplifies incorporating multiple factors (such as a password).

The construction is implemented in C and Python and is open-sourced. This system is fast enough for use on desktop applications with successful authentication usually completing within .30s.

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 physiological, environmental, and sensor noise make it extremely unlikely that w' is 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 a major risk to privacy.

Key derivation from noisy sources An alternate approach derives stable cryptographic keys from noisy data. A *fuzzy extractor* [DRS04, 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.

Iris Key Derivation The work of Hao et al. [HAD06] is probably the most well known work that derives keys from the iris. Their construction consists of an iris image processing transform and a fuzzy extractor using an error-correcting code tailored to errors that occur in the iris. [1] The iris image processing produces a 2048 bit $\{0, 1\}$ vector. They use a degrees of freedom argument to estimate this vector is believed to have 249 bits of entropy [Dau04]. They build an error correcting code that recovers when approximately 27% bit differ.

They claim a key size of 140 bits and security of 44 bits. We claim that their security analysis is incorrect: neither of these numbers correspond to any meaningful security of a fuzzy extractor. We investigate each claim in turn.

Most fuzzy extractors use error-correcting codes to correct noise (our construction does not). Hao et al. use a codeword from an error correcting code with 2^{140} codewords as a one-time pad for the value w. This is known as the code-offset construction. However, security of a fuzzy extractor is *not* the number of keywords of the error-correcting code. The security of the system after error correction is the starting entropy of the biometric minus the number of bits needed for error correction (Dodis et al. [DORS08, Lemma 3.1]). Using this calculation on Hao et al.'s construction yields remaining security of:

$$\begin{array}{rl} 249 & \text{estimate on the entropy of the iris} \\ \underline{-1908} & \text{size of error correcting information} \\ = 0 & \text{bits of security} \end{array}$$

The number 140 would be accurate if iris were truly uniform bits, that is $140 = 2048 - 1908$. However, Hao et al. note the iris has 249 bits of entropy.

Hao et al. [HAD06, Section 4.3] then claim security of 44 bits. They argue an adversary providing a random iris would succeed with probability 2^{-44}. This corresponds to an *online* adversary that does not have access to any stored error-correcting information (does not see p). Fuzzy extractors are designed to improve security against offline adversaries that see p. Designing fuzzy extractors is simple against an online adversary: set p equal to the original point w. This analysis does not corresponds to any meaningful security guarantee.[2]

It is possible that this scheme is secure "in practice." The iris image processing system used in the system (developed by Daugman [Dau04]) was not released, inhibiting cryptanalysis. Furthermore, public iris image processing systems have worse parameters making the design of a proxy system impractical.

Reusability We describe other related work in Section 8. Most fuzzy extractors provide no security guarantee if a user enrolls their biometric with multiple servers [Boy04, STP09, BA12, BA13]. Secure reuse is more important for biometrics than passwords [BHVOS12, Pg. 564]. 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 $j \neq i$.

[1] 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].

[2] Their claim is slightly different, they ask how frequently an adversary sampling a uniform 249 bit string w' would get a w' within the error-correcting radius of the error-correcting code of the original point. This version of the claim is designed to account for the non-uniformity of the iris distribution. However, the adversary still does not use the error-correcting information to inform their guess. So this still corresponds to an *online* adversary.

1.1 Our Contribution

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 a 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. Canetti et al. present asymptotic analysis that shows error resilience is possible for error rates that are sublinear in the length of the iris transform. The transformed binary strings resulting from two images of the same iris differ in between $10\% - 32\%$ of bits (depending on the transform).

To overcome this gap in error rates, we modify the iris processing and cryptography. Our construction feeds additional information from iris processing to the fuzzy extractor coupling the two stages. In addition, we reduce storage by more than 50% (Section 7). The most important features of the system are:

Security We bound security at 45 bits. We explicitly state conditions needed for this security in Claim 1. This bound assumes that the adversary can create and hash nonuniform bit strings distributed in the same way as processed iris bits in unit time.

We are aware that 45 is not cryptographically secure. The Bitcoin network currently computes $\approx 2^{65}$ hashes every second [bit17]. 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. We view our system as a building block in a secure system like password hashing. In addition, this system may be used as a single factor in a multi factor system. We discuss some thoughts on a multi factor system in Appendix C.

Reusability Our construction allows 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. This has been a limitation of prior work (see Section 8).

Implementation Our construction has been implemented in Python and C. The Python implementation is conceptually simple but does not achieve satisfaction performance in terms of running time or necessary storage.

We report on performance numbers from the C implementation. For this implementation, if data structures are stored in memory, Gen takes 6.40s on average and Rep takes .54s on average (Rep takes at most .30s seventy percent of the time). The data structures for in memory storage consume roughly 200MB.

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]. This dataset is superset of the NIST Iris Challenge Evaluation Dataset [PBF$^+$08].

1.2 Organization

The remainder of this work is organized as follows: in Section 2 we provide background, describe parameters of a basic scheme in Section 4, optimize for security in Sections 5 and 6, describe our implementation in Section 7, discuss additional prior work in Section 8 and conclude in Section 9. Since our security argument builds on statistical analysis security discussion is throughout this work, formal discussion is in Section 6.3.

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.

3 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 $\{0,1\}^n$. 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 *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 We use computational fuzzy extractors in this work [FMR13]. Dodis et al. provide similar definitions for information-theoretic fuzzy extractors [DORS08, Sections 2.5–4.1].

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.*
- Correctness: *if* $\mathsf{dis}(w, w') \le t$ *and* $(r,p) \leftarrow \mathsf{Gen}(w)$*,*

$$\Pr_{p,\mathsf{Rep}} \Pr[\mathsf{Rep}(w', p) = r] \ge 1 - \delta.$$

- Security*: for any distribution $W \in \mathcal{W}$, where* $(R,P) \leftarrow \mathsf{Gen}(W)$*,*

$$\delta^{\mathcal{D}_{s_{sec}}}((R,P),(U_\kappa,P)) \le \epsilon_{sec}.$$

Reusable Fuzzy Extractors A reusable fuzzy extractors [Boy04, CFP$^+$16] supports multiple enrollments of the same biometric. The algorithm Gen may be run multiple times on correlated readings $w_1, ..., w_\rho$ of a given source. Gen produces pairs $(r_1,p_1), ..., (r_\rho, p_\rho)$. 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 $j \ne i$.

Definition 4. *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. The only way to obtaining any information about the "locked" value is guessing the key.

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

Digital lockers can be constructed from variants of the Diffie-Hellman assumption [CD08]. Our construction assumes that HMAC-SHA512 can be used to construct digital lockers. Let H be HMAC-SHA512. The "locking" algorithm outputs the pair

$$\mathsf{nonce}, H(\mathsf{nonce}, w) \oplus (0^\lambda || \mathsf{Key}),$$

where nonce is a nonce, $||$ denotes concatenation, 0^λ is the all zeros string of length λ, a security parameter. Unlocking proceeds by recomputing the hash and checking for a prefix of 0^λ. If this prefix is found then the suffix Key' is output.

Security is defined via virtual grey box obfuscation which says that an adversary when provided with the lock succeeds similar probability as an unbounded simulator provided with oracle access to the functionality but a limited number of queries [BC10].

When the entropy of w is superlogarithmic and the construction is secure, the adversary has negligible probability of finding the correct Key; and if the adversary doesn't find the correct key, then the adversarial information about either w or Key does not change much. 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].

3.1 Sample-then-lock

Our cryptographic construction builds on the reusable fuzzy extractor of Canetti et al. The high level idea is to encrypt the same key multiple times using different subsets of w. Pseudocode for the algorithm is below:

```
Gen($w$):

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

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

  1. For $i = 1, ..., \ell$:
       (i) Set $c_i = \mathtt{Hash}(h_i, w'_{j_{i,1}}, ..., w'_{j_{i,k}})$.
      (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. 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. We use HMAC-SHA512 as our lock algorithm. HMAC must function as a digital locker as long as the input subsets $w_{j_i,1}, ..., w_{j_i,k}$ have high entropy.

A single digital locker requires storage of 64 bytes for the output of the hash 32 bytes for each hash key h_i and storage of each set of locations. Storage costs for location sets depend on the number of positions sampled k. Since the input iris size is 32768 we need 15 bits to describe a location in the iris and thus $15 * k$ bits to describe the set of locations. We discuss efficient storage of location sets in Section 7.

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.[3]

3.2 Transforming the Iris

Image processing techniques are applied to an iris before authentication. Iris image processing is an entire field [BHF08], we do not attempt to summarize it here. We note any technique that produces a binary vector with Hamming errors (fraction of bits that are the same).

We use open-source OSIRIS package. We use this package as a starting point as it is open source and uses representative techniques. OSIRIS takes a standard near infrared iris image and produces a 32768 bit vector w. The stages of OSIRIS are:

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 yielding a complex values at locations (r, θ). This produces a 64×512 vector of complex valued numbers.

4. Binarization: Complex numbers are quantized based on sign to produce two bits.

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].

4 Basic Implementation

Designing a fuzzy extractor is a balance between correctness and security. Roughly, as error tolerance increases, security decreases. The goal is to make the construction match the information theoretic tradeoffs as closely as possible [FRS16]. For sample-then-lock the two tunable parameters are ℓ the number of lockers and k the size of each locker. Increasing k improves security but hurts correctness, increasing ℓ improves correctness but costs time and storage. The two parameters are related by

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

[3]Asymptotic notation is not well defined for a single distribution W. However, we show that the 32% error rate observed in irises is not correctable by this construction.

The starting point 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%.[4] 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 2×10^7 digital lockers. We describe a method for reducing storage costs in Section 7.

In addition, the digital locker security requires the entropy of the subset to be high not just its size. It is sufficient for subsets of w to have high entropy. We show that sampling random iris bits produces high entropy subsets (see Figure 2).

The rest of this work is dedicated to overcoming these problems. We will set the number of lockers $\ell = 10^6$. As we will see storing the HMAC information and subsets costs approximately 200B per locker for overall storage of ≈ 200 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.

We start by implementing the sample-then-lock construction using the 32768 bit iriscode output. All statistical analysis is performed using the ND-0405 dataset [BF16] which is a superset of the NIST Iris Challenge Evaluation Dataset [PBF+08]. 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.

Figure 1 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/different* comparisons corresponding to the same iris and *intraclass/same* 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 iris 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 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 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 Having established a baseline error rate and estimate of iris entropy our next step is to ask how much entropy does a sample of iris bits have? In the worst case, sampling only preserves the entropy rate of a

[4]The security/correctness tradeoff of our system immediately improves with an iris transform with lower error rate.

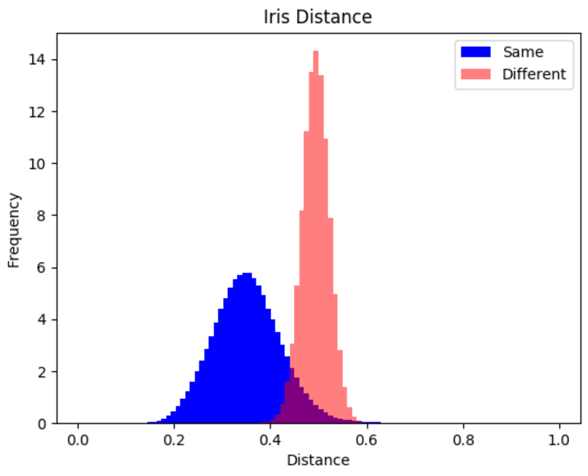


Figure 1: Distribution of distances for the data set.

distribution which for OSIRIS is $311/32768 \approx 1\%$.

Iris entropy is believed to be geographically distributed throughout the iris. The OSIRIS output is produced by convolving a fixed Gabor filter at overlapping regions of the unwrapped iris. So one would expect nearby bits to be correlated. If only 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 distinct positions (between 0 and 32767).
2. Compute the intraclass and interclass histograms for the dataset restricted to these positions.
3. Compute the μ and σ for the interclass histogram. (Using the same method as in Figure 1).
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. 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. Indeed we find these two calculations, the average of entropies and the *average min-entropy*, differ substantially. In some rare events, positions are chosen close together yielding a low entropy.

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 2. We note that the entropy rate is significantly higher than the worst case of 1%. At some points in Figure 2 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 $k = 32$.

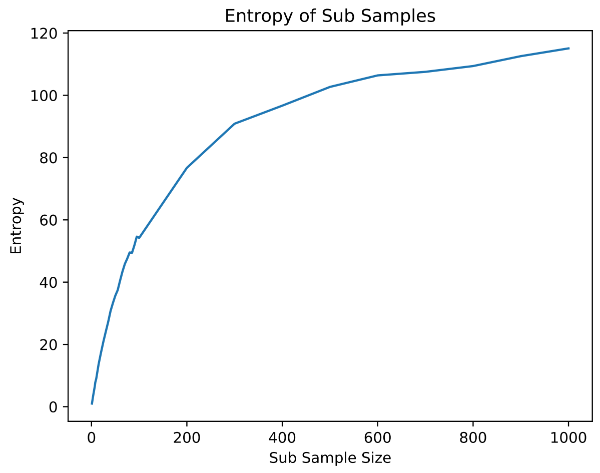


Figure 2: In the worst case subsampling only preserves error rate. For the iris, subsampling greatly increases entropy rate from 1% to over 80%.

3. Interpolate between $k = 30$ and $k = 35$ on Figure 2 to obtain that entropy/security $e = 26$ when $k = 32$.

Thus, starting from the output of the OSIRIS transform gives a security estimate of 26 bits of security. The rest of this work is improving this parameter.

5 Masking unreliable bits

A technique commonly used to improve iris transforms is called *masking*. (Bowyer et al. survey 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]. (We present heatmaps for masking probability and error rate in Appendix A.)

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.

6. Compute $\mu_{thres,i}, \sigma_{thres,i}$ for trial i.

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 1: 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 Thres.	# 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 2: Security of *sample-then-lock* when restricting to bits that exhibit lower error rate across the data corpus.

7. Compute the entropy $e_{thres,i}$ for trial i.
8. Compute the overall entropy as $e_{thres} = -\log \mathbb{E}_i\, 2^{-e_{thres,i}}$

The outcome of this analysis is in Table 1. The goal of using masking information was to determine if there were global locations that have higher error rate.

Using 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. 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 $thres \in \{1, .99, .98, .97, ..., 0\}$.

There is a strong correlation between these locations and both methodologies exhibit similar entropy at their optimum. Both methods allow 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.

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 and entropy 32.
Note: The reader may notice that the top entries in Tables 1 and 2 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 numbers differed by .3, the difference of 1 is due to rounding.

6 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, $\mathtt{sign}(a_i)$ and $\mathtt{sign}(b_i)$. 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.

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 captured 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 satisfactory false accept and false reject rates without such information. Our analysis only refers to a_i because we are using a transform where b_i is discarded. In order to use $|a_i|$ in our system, we must show the following properties:

1) Utility: There exists a negative correlation between $|a_i|$ and error rate in $\mathtt{sign}(a_i)$ in subsequent readings.

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

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

The next subsections demonstrate these properties.

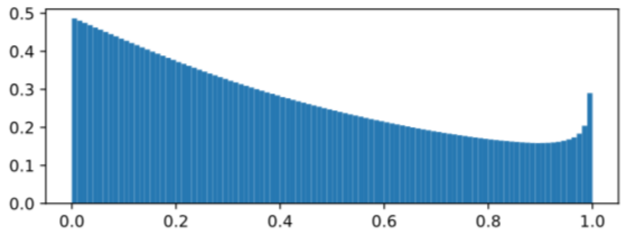
6.1 Utility

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

$$\Pr[\mathtt{sign}(a_i)' \neq \mathtt{sign}(a_i) | |a_i| \in [ca, cb]] < \Pr[\mathtt{sign}(a_i)' \neq \mathtt{sign}(a_i)].$$

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.


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

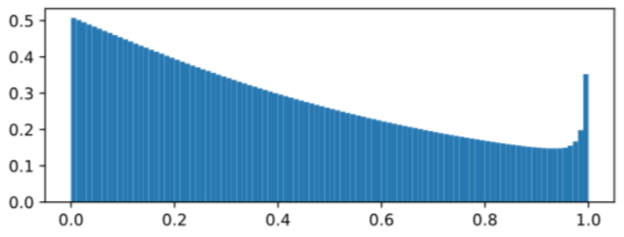

(b) Negative values: Histogram of confidence vs. error rate.

Figure 3: Histogram of correlation between error rate (y-axis) and magnitude of confidence information. Nonnegative confidence values are on the left histogram. Negative confidence values are in the right 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.

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 3. 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?

6.2 Correctness

The lock-then-sample construction needs many bits in the source to create many unique and dissimilar subsets. The construction computes 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).

Subsample Size	64	128	256	512	2^{10}	2^{11}	2^{12}	2^{13}
Correctness (%)	17	41	52	56	60	60	60	60

Table 3: 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. Analysis performed using Python implementation described in Section 7.

We now determine the minimum number of bits that must be fed to *sample-then-lock* to maintain correctness. This analysis was performed using an actual fuzzy extractor not just statistics. For clarity we defer discussion of the implementation until the next section. 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.
3. Run Gen with the restricted w.
4. Execute Rep on all images of the same iris (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 3. 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. This analysis required substantial computational power and restricting trials was necessary to keep the computation tractable.

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 3. 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 4: 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 4. 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 tend to be in close proximity and thus are redundant or ii) 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 4. 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.

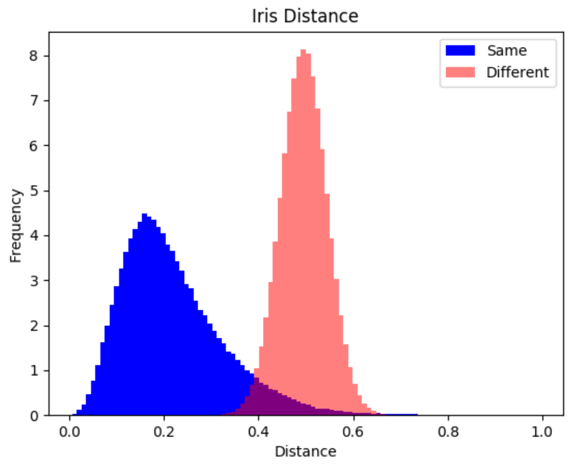


Figure 4: Distribution of distances for the data set restricted to high confidence bits. This figure is formed by selecting all locations of an image that have magnitude between $[343, 2132]$ or $[-2132, -307]$ and then computing the fractional Hamming distance with all other images in the data set restricted to these locations.

6.3 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 $(W, \mathcal{I})$ be a pair of random variables corresponding to sampling an iris and the location of that iris's high confidence bits. 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:

Claim 1. *If*

1. *Assumption 1 holds,*
2. *The average min-entropy of high confidence bits is 45, denoted $\tilde{H}_\infty(W_\mathcal{I}|\mathcal{I}) \geq 45$, and*
3. *HMAC-SHA-512 serves as a composable digital locker*

then the construction is a reusable fuzzy extractor, an ideal adversary requires q queries to the underlying locker to break a locker with probability $\leq \frac{q}{2^{45}}$.

The security of the digital locker is defined by the number of queries that must be made to the hash function. Digital lockers are defined using the real-ideal paradigm and it is difficult to state the precise difficulty of breaking a concrete implementation. Above we state security of breaking the security in the ideal world. Given the security levels considered in this work, we do not expect HMAC-SHA256 to be the weakest link of the implementation.

There are tools to quickly generate guesses according to the password distribution but we are not aware of comparable tools for the iris. Thus, generating a 65 bit long string distributed according to the iris may take more time than generating a random string of 45 bits. Naturally, these tools may be developed in response to widespread deployment of cryptographic iris authentication.

Unlinkability 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 as the only stored information is distributed as random hash outputs.

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.

7 Implementation

We implemented our construction in both Python and C and both implementations are open sourced [FSS18]. Previous implementations of fuzzy extractors required expertise in error-correcting codes, forcing the user to understand finite fields. Our construction only requires repeated evaluation of a hash function.

7.1 Python

Our discussion of the Python library is intended to demonstrate the simplicity of our construction. The Python library is largely unoptimized. The entire Python library is 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
```

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 $j_{i,1}, ..., j_{i,k}$.

Similarly, a single threaded version of Rep is conceptually clear.

```
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
```

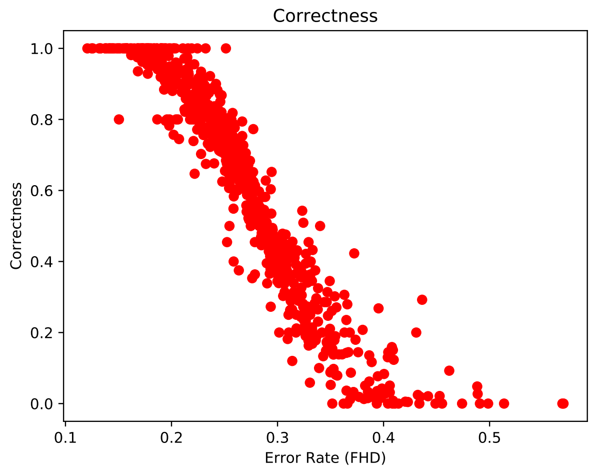


Figure 5: Correlation between correctness of Rep and the error rate of an individual's eye. The mean error rate across the corpus is 60%. Based on subset size of 43 bits with the 12000 bits that have probability of being masked of at less than 5%.

The Rep functionality is embarrassingly parallel. We implementation a parallel version that simply partitions the hashes to be performed. Rep succeeds when one of these threads returns. Unfortunately, neither implementation is fast enough for a real time authentication system. In the Python implementation Gen takes 220s. We implemented a parallel version of Rep which takes 12s. Since Rep must be performed on every authentication this is not fast enough for most use cases. In addition, for the confidence version of the scheme storage of 96 bytes for the HMAC information, each subset requires 15 locations bits to index a single location with 65 locations yielding $15*65/8 = 122$ bytes required for each subset. This yields storage of 218 MB for each enrolled iris. These performance numbers do not include disk read time, which was greater than the computation time.

To improve on these numbers we implement an optimized albeit more complex C version of the library. Before moving to that implementation, we evaluate whether the expected correctness is observed on actual data.

Correctness Evaluation The prior sections argue security for a fixed correctness level. While it is difficult to assess the "true" security of the implementation it is possible to assess the empirical correctness of the system. This performance analysis was performed on a Dell Precision Tower 7000 Series with a 4 Xeon E5-2620 v4 processors and 64GB of RAM. The computation was parallelism bound. We tested correctness across the data corpus for the masking and confidence versions of the implementation. 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 higher at 60%. As expected the correctness is highly correlated with the error rate of the underlying iris. This correlation is demonstrated in Figure 5.

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 6.

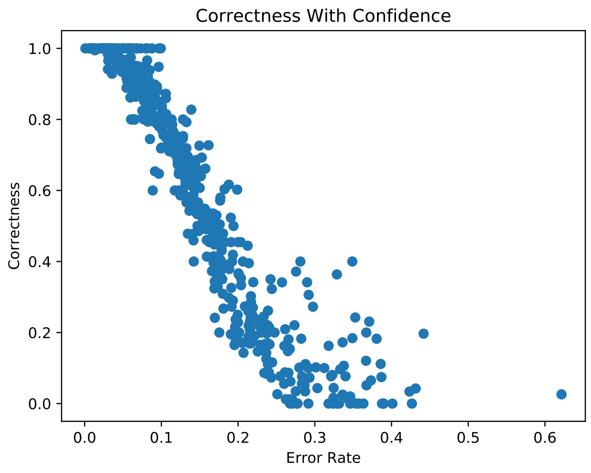


Figure 6: Correlation between correctness of Rep and the error rate of an individual's eye. The mean error rate across the corpus is 60%. Based on subset size of 81. Subsets selected by first 1) removing bits that are likely to be masked and 2) only selecting from bits in high confidence range.

7.2 C

We also developed an optimized C implementation designed for fast Rep performance. As this operation is used at every authentication its speed is more important than Gen which is only used when a user enrolls with a new service. For this implementation we used Libsodium as the cryptographic backend and use HMAC-SHA-512 for the hash function. This library makes use of low level bit level operations for quickly packing and selecting bits the iris vector. In preliminary testing a major obstacle to fast Rep was disk load time. Recall, each p is 218MB if perfectly packed (no serialization overhead).

Reducing Storage We make a crucial observation about *sample-then-lock*:

Security of sample-then-lock does not depend on the subsets j_i being chosen independently.

Canetti et al. [CFP+16, Section 4] note that rather than using independent subsets they could be selected using a sampler [Gol11]. In fact the security argument holds if all subsets are exactly the same.

Theorem 1. *Let λ be a security parameter, Let $\mathcal{W}$ be a family of sources where for a random subset $\mathcal{I}$, $\tilde{\mathrm{H}}_\infty(W|\mathcal{I} \geq \alpha$ for $\alpha = \omega(\log \lambda)$. Then for any $s_{sec} = \mathtt{poly}(\lambda)$ there exists some $\epsilon_{sec} = \mathtt{ngl}(\lambda)$ such that* sample-then-lock *is a $(\mathcal{Z}^n, \mathcal{W}, \kappa, t)$-computational fuzzy extractor that is $(\epsilon_{sec}, s_{sec})$-hard with error $\delta = \mathtt{negl}(\lambda)$.*

The construction is secure each individual $\mathcal{I}_j$ is uniformly distributed even if $\mathcal{I}_{j_1}$ and $\mathcal{I}_{j_2}$ are arbitrarily correlated. (No claim about correctness is made in this case.)

Proof Sketch. Here we do not argue correctness. We sketch why security holds even if the subsets are dependent. Crucially, we still require their marginal distribution to be uniform.

The original proof of Canetti et al. first transitions to an "ideal locker." They then show that given q queries to any of the lockers, the probability of any query unlocking a locker is bounded above by $q(q+1)/2^\alpha$. This argument is in the worst case: 1) it assumes the adversary wins if they get a response other than $\perp$ from any locker and 2) it considers the query response string for the best adversary. They make this argument by showing that entropy decreases by at most the support size of possible query responses [DORS08, Lemma 2.2b]. Since entropy is consider 0 with a single match response the number of query responses can be bound by $q+1$ regardless of the number of lockers. Putting these facts together, we can recover the full argument as long as the probability of opening any locker begins at $2^{-\alpha}$. This is guaranteed by the fact that each subset is uniformly chosen (though subsets may be dependent). □

This theorem gives us a mechanism for saving on storage size. However, using the same subset for each locker will destroy the correctness argument.

We change how subsets are sampled while ensuring each individual subset is uniform. The new scheme works as follows:

1. Choose a master subset $\mathcal{I}$ uniformly at random
2. For each locker j generate a permutation $\pi_j : \{0,1\}^{32768} \rightarrow \{0,1\}^{32768}$.
3. Apply π_j to each element of $\mathcal{I}$ to get $\mathcal{I}_j$.

Rather than storing each subset we can now store the master subset and the permutations. To keep overall storage we use CHACHA20 with input permutation number j as the permutation. Specifically, we

1. Select a single master CHACHA20 key
2. Encrypt the permutation number j, creating $15 * 32768$ bits of output c.
3. We split c into fifteen bit sections $c_1, ..., c_{32768}$.
4. The $\pi_j(i) = c_i$

In our masking implementation we average 6000 confident bits so rather than generating a permutation from $\{0,1\}^{32768} \rightarrow \{0,1\}^{32768}$ we can generate a much shorter permutation with ciphertext output of $\approx 13 * 6000$. This reduces overall storage to a single CHACHA20 key, the single randomly generated subset, and the HMAC information for each locker. This reduces storage from 218 MB to 96MB. All of this information is assumed to be public.

Generating these permutations takes additional computation but allows a smooth tradeoff between storage cost/speed and computation time. In our performance numbers we separate out this operation and call it *Subset Generation.*

Performance Here we report on the speed of our C implementation. We consider the speed of three different operations, Gen, Rep and subset generation. We do not include time for subset generation in Gen and Rep. Furthermore, we do not include disk read time. The reported times for Rep assumes the data structure is already in memory. Depending on the use case the data structure for p may be stored in memory, on disk, or regenerated as needed. Importantly, subset generation does not depend on the private biometric and can be performed ahead of time (e.g., prior to an employee starting their shift).

On average, Gen takes 6.40s, Rep takes .54s, and subset generation takes 12.93. This data is averaged across 100 runs. The time for Rep depends on how quickly an opening locker is found. Roughly, Rep is slow when it fails to open and has to wait for all threads to finish. The distribution of Rep times is in Figure 7. Over 72% of the time Rep completes in less than .30.

8 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].[5] Boyen showed a simple coding construction that matched this bound. There were not many constructions even in this weak model [ACEK17]. Applied works showed that many fuzzy extractors were not reusable [STP09, BA12, BA13], meaning that the negative result of Boyen was not only a theoretic issue.

Recent work considers computational security [FMR13] in part to sidestep this result. Alamelou et al. construct a reusable fuzzy extractor is for a different distance metric (set difference metric), their system is not applicable

[5]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.

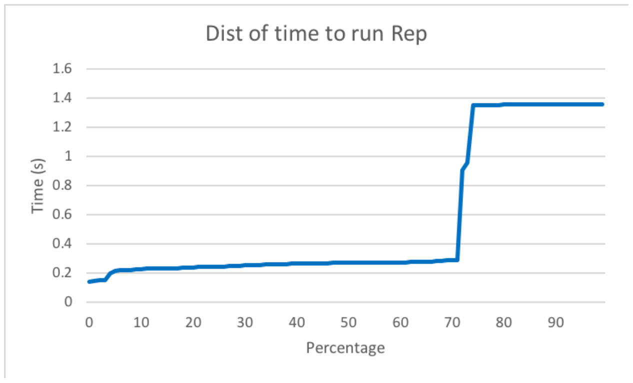


Figure 7: Cumulative distribution of time to run Rep for C implementation. Does not include time to regenerate subsets or load from hard disk. The "inflection point" is samples that did not unlock as they had to wait for all threads to finish all processing all lockers.

Scheme	Limitation
Code Offset [Boy04]	Assumes noise uncorrelated to iris
LWE Decoding [ACEK17]	Assumes noise uncorrelated to iris
Diffie-Hellman [WLH18]	Assumes noise (largely) uncorrelated to iris
Pseudo. isometry [ABC+18]	Applicable for set difference metric
Sample-then-lock [CFP+16]	Sublinear correction capacity

Table 5: 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. The Diffie-Hellman construction requires a small amount of correlation, it is secure if each w_i has high entropy conditioned on the differences between all pairs of readings.

for the iris [ABC+18]. 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 5.

Iris Key Derivation Systems The work of Hao et al. [HAD06] is covered in the introduction. 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 4). 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.

9 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. In addition, the necessary assumptions for security are clearly articulated in Claim 1.

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

Scheme	Security	Subset Size
Basic	26	32
Masking	32	43
Confidence	45	65

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.

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.

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 a fact of life. This work explores how secure cryptographic techniques can be made for real biometrics. While our system does not achieve "cryptographic" security levels, we believe they are 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+18] Quentin Alamélou, Paul-Edmond Berthier, Chloé Cachet, Stéphane Cauchie, Benjamin Fuller, Philippe Gaborit, and Sailesh Simhadri. Pseudoentropic isometries: A new framework for fuzzy extractor reusability. In *AsiaCCS*, 2018.

[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.

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In *Advances in Cryptology–CRYPTO 2010*, pages 520–537. Springer, 2010.

[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.

[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.

[bit17] Bitcoin difficulty and hashrate. `https://bitcoinwisdom.com/bitcoin/difficulty`, 2017.

[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.

[DRS04] 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*, pages 523–540. Springer, 2004.

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

[FRS16] Benjamin Fuller, Leonid Reyzin, and Adam Smith. When are fuzzy extractors possible? In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 277–306. Springer, 2016.

[FSS18] Benjamin Fuller, Sailesh Simhadri, and James Steel. Computational fuzzy extractors. `https://github.com/benjaminfuller/CompFE`, 2018.

[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.

[Gol11] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 302–332. Springer, 2011.

[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.

[PBF$^+$08] P Jonathon Phillips, Kevin W Bowyer, Patrick J Flynn, Xiaomei Liu, and W Todd Scruggs. The iris challenge evaluation 2005. In *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, pages 1–8. IEEE, 2008.

[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.

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

[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.

[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.

[WLH18] Yunhua Wen, Shengli Liu, and Shuai Han. Reusable fuzzy extractor from the decisional diffie–hellman assumption. *Designs, Codes and Cryptography*, Jan 2018.

[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.

A Additional Statistical analysis

In this section we provide additional statistical analysis. Figure 8 shows the locations that were most frequently masked. In Figure 8, 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 $100 \leq \theta \leq 200$.

Figure 9 shows the error rate of locations in the same polar coordinates as Figure 8.

Our analysis of the security of masking in Table 1 was abbreviated so we provide the full table in Table 7. 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 4 (interclass distribution from Figure 1).
2. When restricted to highly confidence bits of an image as described in Section 6.

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

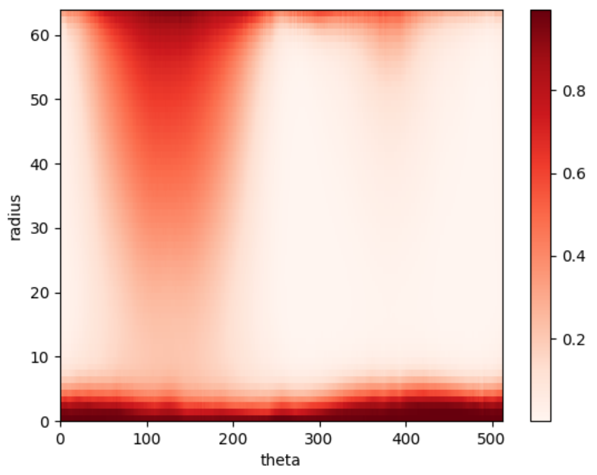


Figure 8: Heatmap of polar coordinate position in the iris (r, θ). The intensity of the pixel indicate the probability that the bit will be "masked" by the transform.

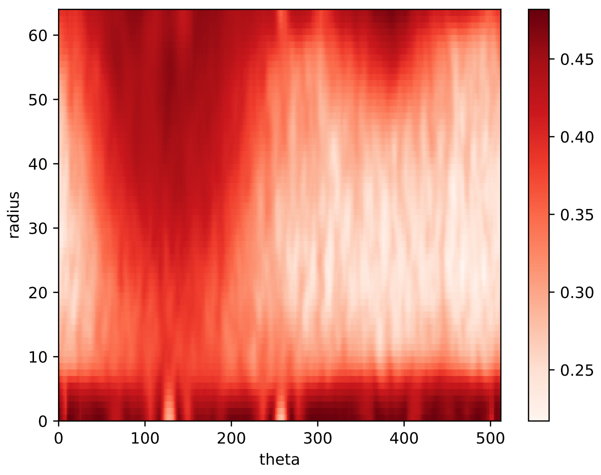


Figure 9: Heatmap of polar coordinate position in the iris (r, θ). The intensity of the pixel indicate the probability that the bit will have an error in a subsequent reading. Note that the scale is different in Figure 8 and this figure.

B Recent Advances in Obfuscation

The recent work of Wichs and Zirdelis [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 $1 \le j_{i,1}, ..., j_{i,k} \le |w|$ 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

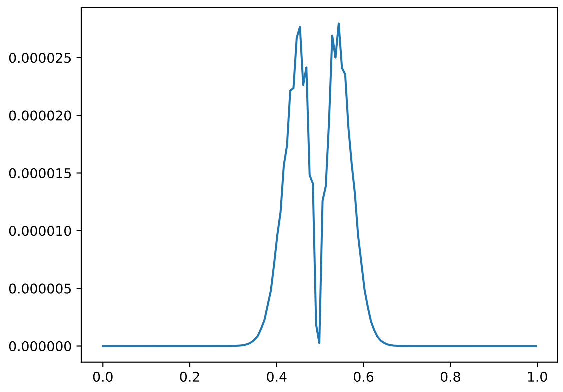


Figure 10: The statistical distance between the interclass comparisons of the base data set and a binomial distribution with the same mean and variance. The x-axis is the Hamming distance (resp. the normalized value of the binomial distribution). The y-axis is the statistical distance between the two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

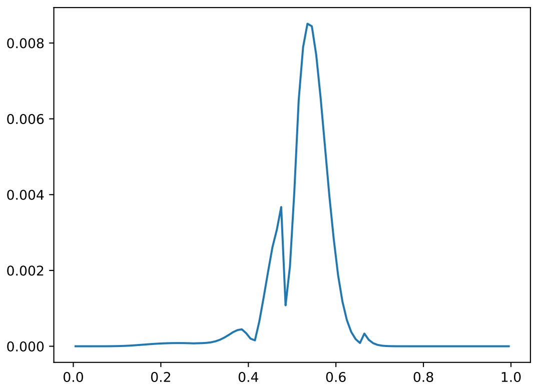


Figure 11: The statistical distance between the interclass comparisons restricted to "confident" bits compared to a binomial distribution with the same mean and variance. The x-axis is the Hamming distance (resp. the normalized value of the binomial distribution). The y-axis is the statistical distance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

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 Zirdelis 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 Zirdelis 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.

C 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, the 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].

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 Table 1.