2007

Forward Secure Fuzzy Extractors

David Goldenberg
College of William & Mary - Arts & Sciences

Recommended Citation

Goldenberg, David, "Forward Secure Fuzzy Extractors" (2007). *Dissertations, Theses, and Masters Projects.* Paper 1539626862.
https://dx.doi.org/doi:10.21220/s2-5j1e-w237

FORWARD SECURE FUZZY EXTRACTORS

A Thesis

Presented to

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Sciences

by

David Goldenberg

2007

APPROVAL SHEET

This thesis is submitted in partial fulfillment of
the requirements for the degree of

Master of Science

[illegible]

David Goldenberg

Approved by the Committee, May 2007

[illegible]

Committee Chair
Assistant Professor Moses Liskov, Computer Science
The College of William & Mary

[illegible]

Assistant Professor Qun Li, Computer Science
The College of William & Mary

[illegible]

Assistant Professor Haining Wang, Computer Science
The College of William & Mary

To everyone who helped me out.

Table of Contents

ACKNOWLEDGMENTS

I acknowledge all the people who helped me complete my thesis. My advisor, my parents, my friends, everyone. Moreover, I acknowledge that if you are reading this, then my thesis is in fact done. I think that is definitely worth an acknowledgement or two.

ABSTRACT

Many cryptographic tools rely on the existence of a secret key. A fuzzy extractor is a cryptographic tool which allows for cryptographically secure keys to be extracted from biometric data. This is useful, as there is a link between biometric data and the user. Also, biometric data is easily stored as it is part of the user. The keys generated from biometric data are uniformly random and can be used in cryptographic tools such as encryption functions, signature schemes, or authentication protocols. However, biometric information can be easily recorded/retrieved by a malicious adversary which means that the resulting extracted keys can be stolen. This is different from other key generation methods, where the key is merely remembered, or stored in some encrypted state. To solve this problem, this thesis applies the idea of forward security to a fuzzy extractor, creating a forward secure fuzzy extractor, an extractor that retains some security even when the adversary gains knowledge about the biometric data of the user. Such a forward secure fuzzy extractor depends on a second factor, and if one factor is corrupted many extracted keys still remain computationally indistinguishable from random.

To create a forward secure fuzzy extractor we define a new primitive, an enhanced extractor. An enhanced extractor functions as an extractor in that for an unknown input of high enough entropy and known random seed the extractor outputs a string which is close to random. However, should the high entropy input become known and the seed remain secret the enhanced extractor functions as a pseudorandom generator. Such a tool may have its own uses independent of a fuzzy extractor.

After constructing an enhanced extractor we show how a forward secure fuzzy extractor can be created out of any enhanced extractor. We also develop an enhanced update function which allows for greater security even when both factors are compromised, and show how we can add robustness to a forward secure fuzzy extractor, even when one factor of our system is revealed.

FORWARD SECURE FUZZY EXTRACTORS

Chapter 1

Introduction

Many cryptographic applications require the use of a secret key for security. This key needs several properties to make it effective.

1. The length of the key needs to be large and the key needs to be randomly generated in order to make a brute force attack infeasible.
2. As keys are often used to distinguish between different users a key needs to have a link to its user.
3. The key should be hard to discover through directed effort (such as theft or other methods).
4. If at all possible, the key should be easy to change. This mitigates the damage an adversary can do by stealing a key.
5. Finally, it should be easy to store and retrieve the key.

The concept of biometric security is based off of the idea that instead of using data an individual remembers, we utilize data that is unique to the user. Such data includes items like a user's fingerprint, iris patterns, or even the user's typing pattern. Any data unique to the user can be used. Because biometric data is thought to be unique to the user, use of biometrics allows for a binding to take place between a user and his key. In addition, the use of biometric data allows for the keys to be stored and to remain hidden easily. Unlike a password, one never forgets their eye or their finger. However, the use of biometrics is not without its issues. The first issue is that biometric data is non-uniformly distributed. Another issue is that biometrics does not immediately allow for strong cryptographic keys. Initially biometrics have been used to provide security by storing a template of a user's biometric, and this template being matched to a later input pattern.[9] However, a template by itself is not a uniform cryptographic key. Moreover, it has been shown that templates and the template matching software may reveal enough information for the biometric data to be approximated from the template.[1] Biometric data is also fuzzy, in that two readings of the same biometric source are usually not identical. This presents a serious problem for cryptologic protocols in that they require the cryptographic key to be exactly entered in order to correctly function.

A fuzzy extractor allows for many of these issues to be resolved. A fuzzy extractor has two parts. The first part takes a biometric reading and produces a public string which can allow for the biometric reading to be recovered. Such a public string is secure in that it reveals 'very little' about the particular biometric reading that it stores. The second part takes a new biometric reading and the public string and outputs a key, where this key is statistically close to random even given knowledge of the public string. Moreover, this second function has an error tolerance. If two readings are close enough to the stored reading in the public string, the fuzzy extractor will output the same key on each reading. [14] Further developments of fuzzy extractors allowed the extractor to be utilized multiple times in the face of a very adaptive adversary while still retaining security and also allowed for public strings to be robust in the face of modification by an adversary.[13, 4]

An problem in using biometric data still unresolved by a fuzzy extractor is that biometric data can be maliciously retrieved by an adversary without regards to the cryptographic system the biometric data is used in. Fingerprints are left on surfaces that we touch, voices and faces can be recorded and analyzed. Should biometric data be stolen, this is a serious theft. Unlike a simple cryptographic key, biometric data cannot be easily replaced. While in other systems the key is only stored inside a person's head or encrypted on disk biometric information is left somewhat 'out in the open'.

Forward security addresses the idea of mitigating the damage any adversary can do by discovering a key. In a forward secure system, the key is changed frequently, with the property that should an adversary steal a key all keys developed before the stolen key still remain secure. This mitigates the damage any adversary can do, as now he cannot decrypt old messages, or try to sign messages using previous keys, or in general perform any operation where a previous secret key is needed. Forward secure symmetric and asymmetric encryption schemes have been created [11, 10] as well as forward secure signature schemes. Forward security is usually accomplished in an asymmetric scheme, where the public key remains fixed, and the secret key iterates from state to state. This iteration of the secret key can even be done if the secret key remains 'encrypted' [5] as is in the case of software like PGP, where a user's secret key is kept encrypted on the computer and unlocked with a password.

To address the problem of an adversary maliciously retrieving biometric data, we apply the idea of forward security to biometrics creating a forward secure fuzzy extractor. A forward secure fuzzy extractor allows for two factors to determine a key, a user's biometric data and a second secret key which stores information about the current state. A forward secure fuzzy extractor allows for the biometric data to be stolen without the resulting exposure of keys. Both biometric data and state information must be retrieved before the correct key can be generated. Moreover, even if both pieces of information are retrieved, keys generated before the corruption of a user's state information still remain secure.

1.1 Our Results

We define and construct a forward secure fuzzy extractor. To construct our fuzzy extractor we define and construct a new tool, an enhanced extractor. Such an extractor functions as an extractor when the biometric reading is unknown and the state is known, but also functions as a pseudorandom generator when the state is unknown and the biometric reading is known. We also show that this is the optimal type of security we can gain as long as we desire the output of the extractor to be longer than the state information. An enhanced extractor cannot offer statistical indistinguishability of its outputs when the biometric reading is known. From an enhanced extractor we create a forward secure extractor, a tool which functions as an extractor and also offers computational indistinguishability of its previous outputs when the biometric reading is known, even when a later random seed is revealed. From a forward secure extractor and a fuzzy sketch we construct a forward secure fuzzy extractor, and give an enhanced update function which allows such an extractor to gain security properties not found in other forward secure primitives. Finally, we list several authentication protocols that utilize fuzzy extractors and examine how a forward secure fuzzy extractor can be incorporated into these protocols.

1.2 Roadmap

In the Chapter 1 we introduce some general concepts that will be utilized throughout the rest of the paper. In Chapter 2 we introduce past work on fuzzy extractors. In Chapter 3 we formally define the concept of a forward secure fuzzy extractor, and in Chapter 4 we give a construction and show how this construction can gain an enhanced security property. In Chapter 5 I give constructions of several authentication schemes that utilize fuzzy extractors. Finally we conclude in Chapter 6.

1.3 Past Work

Fuzzy Extractors were first developed by Dodis, Reyzin, and Smith [14]. There the tool was formally defined and they also gave a proof for how a fuzzy extractor could be formed through the combination of two other tools, a 'fuzzy sketch' and an extractor. An extractor is a function which takes two inputs: a long string sampled from a non-uniform distribution, and a much shorter string randomly generated. Such an extractor outputs a string longer than the short seed which is close to uniformly random. A fuzzy sketch is a function which takes a string, and outputs a public string which does not reveal much about the input string. Later, the input string can be recoved from the public string and any string which is close enough to the input string. Boyen in [4], demonstrated that a fuzzy extractor formed in such a way may not be able to be securely utlilised many times, as the public strings and selected keys may reveal information about keys not known by an adversary. A new security definition, that of a reusable fuzzy extractor was introduced. Later, [13, 7] this method was shown to allow for a dangerous man in the middle attack, where an adversary could modify the public string in such a way to be able to know the key that was computed by the user from this modified string. To deal with this, Dodis et al. proposed a robust fuzzy extractor, which has the property that the public string is resistant to modification by any user who does not know the biometric string used to create the public string.

1.4 Notation and Background

In this section we introduce notations and background information that will be used through the rest of the paper.

Notation:

- w represents a string, usually an instance of biometric or 'fuzzy' data.
- W is the distribution from which w is drawn.
- U_l is the uniform distribution on l bit sized strings.
- $d(w, w')$ is a distance function on two strings w and w' in regards to a specific metric.
- $F(x; X_1)$ is a randomized function F with input x and utilizing randomness X_1.
- We define the min-entropy of a random variable A as

  $H_\infty(A) = -log(max_a Pr(A = a))$.
- We define the statistical difference between two distributions as

  $SD(A, B) = \frac{1}{2}\sum_{\forall v} |Pr(A = v) - Pr(B = v)|$.
- For two random variables W_1, W_2, on a metric space $\mathcal{M}$, we will say the variables are t-close, if $\forall w_1 \in W_1$, $w_2 \in W_2$, $d(w_1, w_2) \leq t$.

(Note: This means that each random variable individually is contained in a ball of size t).

- Likewise, two random variables W_1 and W_2 defined on a metric space $\mathcal{M}$ are t far apart if $\forall w_i \in W_1$, $\forall w_j \in W_2$, we have that $d(w_i, w_j) > t$.

- For two distributions X and Y, we say that they are ϵ close if:

  $SD(X, Y) \leq \epsilon$.

- We define a function α to be negligible if

  $\forall c\ \exists n : \forall N > n\ \alpha(N) < \frac{1}{N^c}$

- A function ϵ is non-negligible if:

  $\exists c\ \exists n : \forall N > n\ \epsilon(N) > \frac{1}{N^c}$

- Often when we speak of negligible or non-negligible functions we will simply denote them as α, or ϵ.

- We define the idea of computationally indistinguishable in the following way:

  Two distributions, A and B are computationally indistinguishable if $\forall$ Turing Machines T where the running time of T is $poly(k)$, $\exists k$:

  $Pr[T(A) = 1] - Pr[T(B) = 1] \leq \epsilon(k)$, for some negligible ϵ. Such a parameter k is known as the security parameter.

- The random oracle is a public oracle where given a specific value in its domain, randomly selects a value in its range as its output and associates that output with the specific input for all future queries. Such an oracles is available to any adversary as well as a user. [3]

Metric Spaces

A metric space is a set $\mathcal{M}$ along with a function d, which has the following properties.

$\forall x, x' \in \mathcal{M},\ d(x, x') \geq 0.$

$\forall x, x' \in \mathcal{M}\ d(x, x') = 0$ if and only if $x = x'$.

$\forall x, x' \in \mathcal{M}\ d(x, x') = d(x', x).$

$\forall x, y, z \in \mathcal{M}\ d(x, z) \leq d(x, y) + d(y, z)$

Error Correcting Codes

For a given metric space $\mathcal{M}$, we can define an *error correcting code* on $\mathcal{M}$. An error correcting code is composed of a set $Code \subset \mathcal{M}$, and two functions, C, and D. C, the encoding function takes elements from some domain K and maps them to elements in $Code$. The decoding function D takes any element $m \in \mathcal{M}$, and maps m to $Code_k$, where $Code_k$ is an element in $Code$ such that $d(m, Code_k)$ is minimized. The *error correcting distance* of an error correcting code is the largest radius t such that for every element $m \in \mathcal{M}$ there is at most one code word in the ball of radius t centered on m. We denote a specific error correcting code by as an $(\mathcal{M}, K, t)$ code. We also define a linear code as a code where the set $Code$ is a vector space.

Chapter 2

Past Protocols and Previous Work

The first protocol we define is a fuzzy sketch. Informally a fuzzy sketch allows for a piece of data to be stored securely in a publicly viewable 'sketch', (a public string which does not reveal 'much' about the stored data), and this data can be recovered as long as data sufficiently 'close' to the original stored data is produced. Formally, a fuzzy sketch is the following [14]:

Definition 2.1 *An* $(\mathcal{M}, m, m', t)$ *fuzzy sketch is a randomized function* $\mathsf{Fsk} : \mathcal{M} \rightarrow \{0,1\}^*$*, that has the following properties:*

- *There exists a recovery function* Rec *such that* $\mathsf{Rec}(w', \mathsf{Fsk}(w)) = w$ *as long as* $d(w, w') \leq t$.
- *For all random variables* W *over* $\mathcal{M}$ *where* $H_\infty(W) = m$*, the average min-entropy of* W*, given* $Fsk(W)$ *is at least* m'*. Namely,* $H_\infty(W|Fsk(W)) \geq m'$.

Below is an example of a fuzzy sketch construction.

Definition 2.2 (Fuzzy Sketch for the Hamming Metric) *The following is a fuzzy sketch for the Hamming Metric. Let C be the encoding function of an error correcting code, and let D be the decoding function.*

- $\mathsf{Fsk}(w; X_1) = w \oplus C(X_1) = P$.
- $\mathsf{Rec}(w', P) = P \oplus C(D(w' \oplus P))$.

For a fuzzy sketch, we would like the sketch to be such that $m - m'$ is negligible[8]. It is impossible for a sketch to bound a loss of Shannon Entropy. For instance, the above fuzzy sketch loses $n - k$ bits of Shannon entropy, where k is the dimension of the error correcting code C. However, it is possible to create an *entropically secure* sketch. Such a sketch may reveal a non-negligible amount of Shannon Information, however the ability for any user to compute a function of the random variable W given the sketch is only negligibly different than a user's ability to compute a function of W without. Formally:

Definition 2.3 (Entropically Secure Sketches) *We say that a probabilistic map $S()$ hides all functions of W with leakage ϵ if $\forall A\ \exists A' : \forall f\{0,1\}^* \rightarrow \{0,1\}^*$:*

$|Pr[A(S(W)) = F(W)] - Pr[A'() = F(W)]| \leq \epsilon.$

S is called m, ϵ entropically secure if S hides all functions of W, $\forall W : H_\infty(W) = m$ with leakage ϵ.

An important thing about entropically secure sketches is that an entropically secure sketch bounds the loss of min-entropy to be quite small. This follows from the fact that the probability of computing the identity function on W, $F(w) = w$ given a sketch is at most ϵ different from computing the identity function given the sketch.

The second protocol we define is an extractor. Informally, an extractor is a function that takes a string generated from a distribution that is 'not-quite' random, and from that generates an output whose distribution is statistically indistinguishable from random. To accomplish this the extractor takes a second input, a small random string X. In a sense an extractor takes a large string that is 'somewhat' random, and extracts the ranomness from it, yielding a smaller string that is completely random. More formally, an extractor is defined as follows [7]:

Definition 2.4 *An (n, m, l, ϵ) extractor is a randomized function* Ext, Ext $: \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^l$ *with the following property:*

For all distributions W where $H_\infty(W) = m$ we have that:

$SD((\mathsf{Ext}(W; X), X), (U_l, X)) \leq \epsilon$.

Finally, we describe a fuzzy extractor. The fuzzy extractor will combine elements of both a fuzzy sketch, and a strong extractor. Namely, a fuzzy extractor will be able to take a string from a distribution with high entropy, as well as a small completely random string, and output a close to random string, which remains statistically close to random even given knowledge of the public string. Formally [14]:

Definition 2.5 *A (m, l, t, ϵ) fuzzy extractor is a pair of algorithms* Gen *and* Rep *which posess the following properties:*

- Gen, *a randomized algorithm on input $w \in W$ outputs a random string $R \in \{0,1\}^l$ as well as a public string $P \in \{0,1\}^*$.*
- Rep, *on input w' and P' outputs R.*
- *These functions have the following properties:*
  - ***Correctness:*** $\forall w, w' : d(w, w') \leq t$, *if* $(R, P) \leftarrow \mathsf{Gen}(w)$ *then* $\mathsf{Rep}(w', P) = R$.

- ***Security:*** *If $H_\infty(W) \geq m$ and $(R, P) \leftarrow \mathsf{Gen}(W)$, then we have that $SD((R, P), (U_l, P)) \leq \epsilon$.*

Notes: It is important to note that a fuzzy extractor construction is dependent on the metric that is to be used. If strings are compared under different metrics, this will result in a different fuzzy extractor construction. Also, the uniformity of the extracted keys holds up linearly under composition. Namely, given public strings $P_i, i = 1, 2, 3, ...k$ and keys $R_i, i = 1, 2, 3, ...k$, an adversary's ability to distinguish the vector of keys R_i from random strings R'_i is at most $k\epsilon$. We will prove this formally later on.

2.0.1 Previous Work On Fuzzy Extractors

In this section we go over the previous papers on fuzzy extractors. In 2004 Dodis, Reyzin and Smith [14], formally developed the idea of the fuzzy extractor as a method for turning biometric information into cryptographic keys. The main theorem of the paper proved that given a strong extractor and a fuzzy sketch,that these could be combined to form a fuzzy extractor.

Theorem 2.6 (Building Fuzzy Extractors) *Given a fuzzy-sketch and an extractor, the following is a fuzzy extractor construction:*

- $\mathsf{Gen}(w; X_1, X_2)$*: Set* $P = (\mathsf{Fsk}(w; X_1), X_2)$*, and set* $R = \mathsf{Ext}(w; X_2)$.
- $\mathsf{Rep}(w', (V, X_2))$*: Recover* $w = \mathsf{Rec}(w', V)$ *and output* $R = \mathsf{Ext}(w; X_2)$.

Proof: Full proof found in [14]. Informally, we can say that due to the properties of the extractor, the statistical difference between R and U_l should be $\leq \epsilon$ thus obtaining the security property. The fact that $\mathsf{Fsk}(w)$ does not reveal 'much' about the public string allows this property to hold even given the public sketch

$\mathsf{Fsk}(w)$. Moreover, the fuzzy sketch allows for the same R to be generated for values close to w obtaining the correctness property. ■

Later, Boyen in [4] would show that while Lemma 2.6 holds for a fuzzy extractor that is used once, it may not hold for a fuzzy extractor utilized multiple times. He develops two new definitions of security, Insider Perturbation Security and Outsider Perturbation Security. Informally, Outsider Perturbation Security reflects the idea that an adversary should not be able to find the biometric string w, even given multiple different public strings P formed by $P = Fsk(w)$. Insider Perturbation Security is a stronger definition and allows the adversary to not only see multiple different public strings P_i, but also can view the result of extracting the random string R_i from a public string of the adversary's choosing and a biometric reading. The adversary succeeds if given a challenge value, he can decide whether or not the value is a key that would be extracted from a string P_i where P_i was returned from an outsider query, or a randomly selected value. It is important to note that Boyen developed constructions for a reusable fuzzy extractor that used a 'reusable sketch'. Boyen created constructions that could allow for any extractor as the reusability property was gained through the fuzzy sketch tool, and then utilized Theorem 2.6. Before we give the formal definition of a reusable sketch we need to explain the idea of an admissible family of perturbations. A perturbation family on a metric space $\mathcal{M}$, D_d, is admissible if $\forall \delta \in D_d, \forall w \in \mathcal{M}, d(w, \delta(w)) \leq d$. We also define the oracle $\mathcal{O}_{reuse}$. Such an oracle acts in the following way.

Pre-Processing

The adversary sends the oracle a distribution W.

The oracle $\mathcal{O}_{reuse}$ samples $w \in W$.

- $\mathcal{O}_{reuse}(1, \delta_i)$.

1. $\mathcal{O}$ creates $P_i = \mathsf{Fsk}(\delta_i(w))$.

2. $\mathcal{O}$ returns P_i to A.

- $\mathcal{O}_{reuse}(2, \delta_i, P')$.

  1. $\mathcal{O}$ computes $\mathsf{Rep}(\delta_i(w), P') = R$.

  2. $\mathcal{O}$ returns R to the adversary.

- $\mathcal{O}_{reuse}(3, P_i)$.

  1. If P_i is not a string that was returned as part of a query $\mathcal{O}_{reuse}(1, \delta_i)$, O returns $\perp$.

  2. If A has ever made a query $(2, \delta', P_i)$ where δ' does not have minimum displacement greater than t, A returns $\perp$.

  3. Otherwise $\mathcal{O}_{reuse}$ randomly selects a bit b. If $b = 1$ then the oracle returns a random string. If $b = 0$ then the oracle returns $\mathsf{Rep}(w, P_i)$ and returns it A.

- $\mathcal{O}_{reuse}(4, \delta_i)$.

  1. If $\mathcal{O}_{reuse}$ has received no queries of the type $(3, P_i)$ O returns $\perp$.

  2. Otherwise $\mathcal{O}_{reuse}$ returns $\mathcal{O}_{reuse}(1, \delta_i)$.

- $\mathcal{O}_{reuse}(5, \delta_i, P')$.

  1. If A has not made query $(3, P_i)$, $\mathcal{O}_{reuse}$ returns $\perp$.

  2. If A has made a query $(3, P')$ and δ_i has minimum displacement less than t, $\mathcal{O}_{reuse}$ returns $\perp$.

  3. If the above two conditions do not hold:

     $\mathcal{O}_{reuse}$ returns $\mathcal{O}_{reuse}(2, \delta_i, P')$.

Now we define a reusable sketch as follows.

Definition 2.7 (Reusable Sketch) $\forall A^{mathcalO}, Pr[A^{\mathcal{O}_{reuse}} \rightarrow b' : b = b'] \leq \alpha(k)$

We can also define a weaker sketch, a sketch reusable vs. outsider queries. The oracle acts in a similar way, except it does not accept queries of the type $(5, \delta_i, P')$ or $(2, \delta_i, P')$. The adversary in the new definition only has to guess at the biometric which is used in one query, and not decide between random keys.

This definition of reusability has its own problem, as written in [13]. There it was shown that since under the definition of Insider Perturbation Security, the adversary knows the value of an extracted string R_i for some public string P_i and biometric w, this allows a man-in-the-middle attack where an adversary replaces a string P_i' with P_i, and now knows the supposedly secret string R_i. Boyen et al. propose a new definition, that of a robust fuzzy extractor. The robust fuzzy extractor is defined as follows:

Definition 2.8 *A robust fuzzy extractor has all the properties of a normal fuzzy extractor, with two additional properties:*

- *The function* Rep *is allowed to return the value* $\perp$.
- *Consider the adversary who gains the two strings R_i and P_i,*

  where $(R_i, P_i) \leftarrow \mathsf{Gen}(w)$ and then outputs $P' \neq P$.

  We say that a fuzzy extractor is robust if:

  $\forall w'$ *where* $d(w, w') \leq t$ $Pr[\mathsf{Rep}(w', P') \neq \perp] \leq \alpha$*, for a negligible* α.

In addition to a robust fuzzy extractor, the paper [7] allows for a keyed robust fuzzy extractor, which is just like a normal robust fuzzy extractor except that it requires a seperate secret key SK. This secret key is used to allow the extraction of a cryptographic key from the same biometric source. A keyed robust fuzzy extractor hides the value of the secret key to an unbounded prover, even given the value of the public string P.

Chapter 3

Forward Secure Fuzzy Extractors

3.1 Forward Secure Fuzzy Extractors

In this section, I define the concept of a forward secure fuzzy extractor. Informally, I would like a forward secure fuzzy extractor to simply add on the forward security property to a fuzzy extractor. This means that corruption of a user's biometric data does not result in a total loss of security. I use the same idea as other forward secure tools, that of updating some secret information from from state to state. This update will be deterministically done from the public string, a biometric input, and the previous state. This second factor is necessary because with the correctness property of a fuzzy extractor the retrieval of biometric data allows any individual to extract to any key as long as they have the corresponding public string.

With this second factor we would like to have the forward security property hold until after both factors have been retrieved. However, consider the situation in which state information is stolen first and later biometric information is stolen.

As long as all public strings are sketched from biometric data that is 'close enough' to the stolen biometric, the adversary can use the past state and the biometric reading to compute the biometric reading that was used in the past and thus is also capable of performing of update from that state on. This situation is similar to the possibility of an adversary stealing the encrypted private key and later the decryption key in Boyen's construction that allows for forward security with an untrusted update [5]. Since the decryption key never changes, it can be used in the past to decrypt the previous keys. So, at best the security we can achieve is two-fold.

- Any adversary must receive both the secret key for a specific state, and a 'good' biometric in order to retrieve any key.
- For any adversary which retrieves both the secret state information, and a biometric reading, all keys extracted from before the adversary had stolen the state information remain unknown to the adversary.

The formal definition of forward security will allow the adversary to have access to an oracle $\mathcal{O}_{FE}$. This oracle needs to have state and the oracle will selec the first state randomly from all possible states before the first query by any adversary. The oracle will remember these past states, and can output the past state information on request. Moreover, this oracle has access to a distribution W, where W has min-entropy at least m. During each state the oracle samples $w \in W$ and will first output a public string P_i formed from w, and then wait for a query. After each query, the oracle will use the biometric w, the public string P_i and the current state to update to the next state. The oracle functions in the following way.

- $\mathcal{O}_{FE}(1) = w_i$, where w_i is the biometric for query i. Informally we call this a **Biometric Corruption Query**.
- $\mathcal{O}_{FE}(2) = St_i$ where St_i is the state information for query i. Informally we call this a **State Corruption Query**.
- $\mathcal{O}_{FE}(3) = \mathsf{Rep}(w_i, St_i, P_i)$ We call this an **Insider Query**
- $\mathcal{O}_{FE}(4)$ The oracle selects $b' \leftarrow \{0,1\}$. If $b' = 0$, the oracle returns $R \leftarrow U_l$. If $b' = 1$, the oracle returns $R = \mathsf{Rep}(w_i, St_i, P_i)$. We call this the **Decide Query**
- $\mathcal{O}_{FE}(5)$ The adversary sends a bit $b \leftarrow \{0,1\}$. We call this the **Guess Query**

Any adversary is allowed to make only one decide query and only one guess query. Next, I define the following notation. Let b_i be the number of the $i'th$ biometric corruption query, and let s_i be the number of the i'th state corruption query. Also, let g be the number of the adversary's guess query.

Definition 3.1 (Forward Secure Fuzzy Extractors) *A forward secure fuzzy extractor, (FSFE), is a family of the following functions.*

- $St_0 = \mathsf{FirstKey}(1^k)$, *a randomized function which generates the initial state.*
- $R = \mathsf{Rep}(w', St, P)$, *a function which takes a biometric data string* $w' \in \{0,1\}^{n(k)}$, *the public sketch* $P \in \{0,1\}^*$, *some state information* $St \in \{0,1\}^{s(k)}$ *and outputs the key* $R \in \{0,1\}^{l(k)}$.
- $St_{i+1} = \mathsf{Update}(w', St_i, P)$, *a function which takes a biometric string* w', *a public sketch* P, *state information* St_i, *and outputs the next state.*

- $P = \mathsf{Gen}(w', St_i)$, *a randomized function which takes a biometric string as input and outputs a public string.*

Correctness

$\forall St_i$, where St_i comes from a sequence of $St_0, St_1, St_2, St_3, ..., St_{i-1}, St_i$ where St_0 comes from $\mathsf{FirstKey}(1^k)$, and for $\forall j \leq i$, St_j comes from $Update(w', St_{j-1}, P)$:

If $P = \mathsf{Gen}(w', St_i)$, $R = \mathsf{Rep}(w', St_i, P)$ and $d(w, w') \leq t$, then

$\mathsf{Rep}(w, St_i, P) \rightarrow R' = R$.

Extraction

$\forall W : H_\infty(W) \geq m$, $\forall w \in W$ if $P = \mathsf{Gen}(w, St)$ and $R = \mathsf{Rep}(w, St, P)$ we have that:

$SD((U_l, P), (R, P)) \leq \epsilon(k)$, where ϵ is negligible.

Forward Security

Here we have two cases:

Case 1:

Here A makes both a biometric corruption and state corruption query.

$Pr[A^{\mathcal{O}_{FE}} \rightarrow b : b = b' \text{and } g < s_1] \leq \alpha(k)$ for negligible α.

Case 2:

Here A makes only one type of corruption query, either a biometric corruption or state corruption query.

$Pr[A^{\mathcal{O}_{FE}} \rightarrow b : b = b'] \leq \alpha(k)$ for negligible α.

3.2 Enhanced Forward Secure Fuzzy Extractors

Forward secure fuzzy extractors as defined above, have a forward security property where keys before the first state corruption query remain secure. In this section we define a new function, an enhanced update procedure. $\mathsf{EUpdate}$ is an update procedure that is not done deterministically from the past state information and public string in that it will take a new sample of biometric information and use that information to update to the next state. For an adversary who has some state information, and yet who does not have the specific string used in a running of the EUpdate function the next state will be unknown to the adversary. This allows a stronger forward security property to be gained. Aside from that, an enhanced forward secure fuzzy extractor has identical properties to a normal fuzzy extractor.

Before we state the formal definition we go over the oracle $\mathcal{O}_{EFE}$. This oracle is very similar to $\mathcal{O}_{FE}$ as it has access to a distribution W and is a stateful oracle. The oracle responds as follows: (Note: the corrupt enhanced update and the enhanced update query can be made in conjunction with any other query).

- **Biometric Corruption Query** $\mathcal{O}_{EFE}(1) = w_i$, where w_i is the biometric used in state i.
- **State Corruption Query** $\mathcal{O}_{EFE}(2) = St_i$. The oracle returns its current state.
- **Insider Query** $\mathcal{O}_{EFE}(3) = \mathsf{Rep}(w_i, St_i, P_i)$.
- **Enhanced Update** $\mathcal{O}_{EFE}(4)$, instead of updating using Update, computes $St_{i+1} = \mathsf{EUpdate}(w^*, St_i)$, where $w^* \neq w_i$
- **Corrupt Enhanced Update Query** $\mathcal{O}_{EFE}(5)$ performs an enhanced update query $St_{i+1} = EUpdate(w^*, St_i)$ where $w^* \neq w_i$, and returns w^*.
- **Decide** $\mathcal{O}_{EFE}(6)$. The oracle selects $b' \leftarrow \{0,1\}$. If $b' = 0$, the oracle returns $R \leftarrow U_l$. If $b' = 1$, the oracle returns $R_i = \mathsf{Rep}(w_i, St_i, P_i)$.

- **Guess** The adversary sends a bit b.

We also need to go through the states where an adversary is not expected to know the extracted key when an adversary makes both a biometric and state corruption query. If an adversary A knows the state information and biometric information for a given query, we do not expect the keys to be secure. With a piece of biometric information and the fact that A receives P_i $\forall i$, we assume that A can compute w_i $\forall i$. This assumption may not always hold true if there is a sketch $P_i = Fsk(w)$ where $d(w, w_i) > t$ and the adversary knows w_i. For such a public string we should not gain w from the function Rec. For all queries before the adversary makes his first state corruption query, the keys in those queries should still be secure in that we would like previous states to be non-computable given a later state. Also, the enhanced update query should, in a way, function as a 'reset' in that if an adversary for all previous queries could distinguish between randomly selected strings and properly generated keys, after an enhanced update the adversary should no longer have this advantage, so for all queries after e_1 the first enhanced update query after the first state corruption query, the keys should be indistinguishable from random keys, at least until the next state corruption query is made. With this information, I define the 'Good Set' GS of queries, as $GS = \alpha : \alpha < s_1$ or $e_i < \alpha < s_k$ where $\forall x < k$, $s_x < e_i$.

Definition 3.2 (Enhanced Forward Secure Fuzzy Extractors) *An enhanced forward secure fuzzy extractor is composed of the following functions:*

- $St_0 = \mathsf{FirstKey}(1^k)$, *a randomized function which generates the initial state.*
- $R = \mathsf{Rep}(w', St, P)$, *a function which takes a biometric data string* $w' \in \{0,1\}^{n(k)}$, *the public sketch* $P \in \{0,1\}^*$, *some state information* $St \in \{0,1\}^{s(k)}$, *and outputs a key* $R \in \{0,1\}^{l(k)}$.

- $St_{i+1} = \mathsf{Update}(w', St_i, P)$, *a function which takes a biometric string* w', *a public sketch* P, *state information* St_i, *and outputs the next state.*

- $P = \mathsf{Gen}(w', St_i)$, *a randomized function which takes a biometric string as input and outputs a public string.*

- $St_{i+1} = \mathsf{EUpdate}(w^*, St_i)$, *a function which takes a previous state, a biometric string, and outputs the next state.*

Correctness

$\forall St_i$, *where* St_i *comes from a sequence of* $St_0, St_1, St_2, St_3, ..., St_{i-1}, St_i$ *where* St_0 *comes from* $\mathsf{FirstKey}(1^k)$, *and:*

$\forall j \leq i$, $St_j = \mathsf{EUpdate}(w', St_{j-1})$ *or* $St_j = \mathsf{Update}(w_{j-1}, St_{j-1}, P_{j-1})$.

If $P = \mathsf{Gen}(w_i, St_i)$ *and* $R = \mathsf{Rep}(w_i, St_i, P)$ *and* $d(w_i, w') \leq t$, $R = R' = Rep(w', St_i, P)$.

Extraction

$\forall W : H_\infty(W) \geq m$, *where* $P = \mathsf{Gen}(w, St)$, $R = Rep(w, St, P)$ *we have that:* $SD((U_l, P), (R, P)) \leq \epsilon$.

Enhanced Forward Security

Again we have two cases:

Case 1:

Here A *makes both a biometric and state corruption query.*

$Pr[g \in GS$, *and* $A^{O_{EFE}} \rightarrow b : b = b'] \leq \alpha(k)$ *for negligible* α.

Case 2:

Here A *makes only one type of corruption query.*

$Pr[A^{O_{EFE}} \rightarrow b : b = b'] \leq \alpha(k)$ *for negligible* α.

3.3 Robustness

Here we state the definition of robustness as it applies to forward security. The definition is basically the same as in a non-forward secure construction. Namely, the adversary after receiving public strings $P_i^j = \mathsf{Gen}(w_i, St_j)$ and keys $R_i^j = \mathsf{Rep}(w_i, St_j, P_i^j)$, should not be able to output a string $P_i'^j \neq P_i^j$ where the function *Rep* does not return $\perp$.

Note, we define w_i^j as the i'th biometric sample under state j. We define R_i^j and P_i^j similarly with regards to keys and public strings.

Definition 3.3 (A Robust Forward Secure Fuzzy Extractor) *A RFSFE is a forward secure fuzzy extractor, with the following additional properties:*

First we allow the function Rep *to return the value* $\perp$.

We consider the oracle $\mathcal{O}_{rob}$. $\mathcal{O}_{rob}$ is a stateful oracle, and also has posssession of a random variable W with high entropy. Before any queries, $\mathcal{O}_{rob}$ selects a random starting state using $\mathsf{FirstKey}(1^k)$.

Any adversary A can make the following queries to $\mathcal{O}_{rob}$.

- $\mathcal{O}_{rob}(1)$ *samples $w_i^j \in W$, and outputs $P_i^j = \mathsf{Gen}(w_i^j, St_j)$, and $R_i^j = \mathsf{Rep}(w_i^j, P_i^j, St_j)$. (Informally we call this an information query).*

- $\mathcal{O}_{rob}(2)$ *stores $St_{j+1} = \mathsf{Update}(w_i^j, P_i^j, St_j)$.*

A forward secure extractor is robust if:

$\forall A$, *$Pr[A^{\mathcal{O}_{rob}} \rightarrow P' : \exists i, j : \forall w'$ where $d(w', w_i^j) \leq t$, $\mathsf{Rep}(w', P', St_j) \neq \perp$ and $P' \neq P_i^j] \leq \alpha(k)$.*

Chapter 4

Construction of an Enhanced Forward Secure Fuzzy Extractor

In this chapter we construct an enhanced forward secure fuzzy extractor. We create a new tool, a forward secure extractor and combine it with a fuzzy sketch to create a forward secure fuzzy extractor. By creating a forward secure fuzzy extractor in this way we can get robustness and reusability 'for cheap', in that we can utilize the results of [4] and [7], which obtain robustness by modifying the fuzzy sketch protocol. This modification can be done by adding an authentication function to Gen, which outputs an authenticated public string. Neither method alters the extractor, so by taking those constructions and modifying the extractor we can add forward security with minimal changes. To create a forward secure extractor we construct an enhanced extractor. Such an enhanced extractor is an extractor for fuzzy data that remains unknown and sampled from a distribution with high enough entropy. When a specific piece of fuzzy data is known however, an enhanced extractor functions as a pseudorandom generator with the seed as input.

To begin, we define a new primitive, an enhanced extractor.

4.1 Enhanced Extractors

Informally, we desire an enhanced extractor to have two properties. It needs to function as an extractor for known state information and an unknown biometric string sampled from a distribution with high enough entropy. When the biometric string is known however, we need the output to be computationally indistinguishable from random. (This will require a relationship between the inputs/output sizes and the security parameter). We would like it if the output could be statistically indistinguishable from random, however as we now show that is impossible.

Theorem 4.1 *Enhanced extractors cannot offer statistical indistinguishability of their outputs when the biometric string is known and when the size of the output of the extractor is larger than the size of the seed.*

Proof: The following exponential adversary can always distinguish the outputs of an enhanced extractor from random. Let EExt be an enhanced extractor taking $\{0,1\}^{n(k)} \times \{0,1\}^{s(k)} \rightarrow \{0,1\}^{l(k)}$.

A, (which knows the biometric string w), queries its oracle which either produces $\mathsf{EExt}(w; St_1)$, or a random string $R \in U_{l(k)}$. A then proceeds to run iterate through different seed values St_i obtaining keys R_i. If for any St_i, $\mathsf{EEx}t(w; St_i) = R$ A outputs pseudorandom, otherwise A outputs random.

Since the seed is smaller than the output of the extractor, seeds taken from $\{0,1\}^{s(k)}$ can only map to at most $2^{s(k)}$ different outputs. However, there are

$2^{l(k)}$ possible outputs. So, there are $2^{l(k)} - 2^{s(k)}$ strings $R \in \{0,1\}^{l(k)}$ which cannot be the output of $\mathsf{EExt}(w, St_i)$, for any given w. So the probability of the random oracle selecting a string R such that there is no state St where $R = \mathsf{EExt}(w, St)$ is equal to $\frac{2^l - 2^s}{2^l} = 1 - \frac{2^s}{2^l} = 1 - \frac{1}{2^{l(k)-s(k)}}$ which is negligibly close to one given the security paramter k. So, with all but overwhelming probability, the random oracle $\mathcal{O}_r$ will select one of these invalid strings. If this occurs, then the exponential adversary who brute force searches through all seeds, will find that no seed S' exists such that $\mathsf{EExt}(w, S') = R_i$ and output random. On the other hand, if R_i was the output of EExt then there exists at least one seed S'' such that $\mathsf{EExt}(w, S'') = R_i$ and the adversary will output pseudorandom. ■

Since we cannot have an extractor that offers statistical indistinguishability when the biometric string used is known to the adversary, we formally define a enhanced extractor as follows:

Definition 4.2 (Enhanced Extractor) *A Enhanced Extractor* EExt*, is a family of functions:*

EExt_k *which takes* $\{0,1\}^{n(k)} \times \{0,1\}^{s(k)} \rightarrow \{0,1\}^{l(k)}$ *and which has the following two properties:*

- *For a distribution W where the min-entropy of W is $m(k)$,* $\mathsf{EExt}_k(w, St)$ *is an $(n(k), m(k), l(k), \epsilon)$ extractor.*

- *For a known biometric string w sampled from a distribution W where $H_\infty(W) \geq m(k)$, and unknown state St,* $\mathsf{EExt}(w, St)$ *is a pseudorandom generator with St as the input.*

The first property of an enhanced forward secure extractor is satisfied by any given extractor, as such an extractor must still offer its extraction property when the seed is known and extractors are capable of scaling to larger and larger outputs (though the random seed, entropy needed, and output size will change according to the parameters of the enhanced extractor). However, with most extractors the size of the needed biometric input is exponential in the size of the needed seed which is inefficient. There is a solution in the random oracle model.

Theorem 4.3 (Enhanced Extractors Utilizing Random Oracles) *Let $\mathcal{O}$ be a random oracle which takes strings of size $n + s$ to strings of size l. The following is an enhanced extractor.*

$\mathsf{EExt}(w, St_i) = \mathcal{O}(w||St_i)$.

Proof: The proof here is fairly trivial. As the outputs of O have min entropy l no matter what the entropy of the input, we gain extraction easily. Moreover, we can have $l > H_\infty(W)$ and still 'extract' randomness from the distribution W. The pseudorandomness of this construction comes from the fact that the state St is unknown. Also, here we can allow the sizes of the inputs/outputs l, s, n to be set arbitrarily. ■

In addition to a construction in the random oracle model we can create a construction in the standard model as follows:

Theorem 4.4 (An enhanced extractor construction in the standard model) *Let Ext_k be a family of extractors which takes $\{0,1\}^{n(k)} \times \{0,1\}^{s''(k)} \rightarrow \{0,1\}^{l(k)}$. Let P_k be a pseudorandom permutation family with key k and let $|k| = s'(k)$. Also have $l(k) > s'(k) + s''(k)$. We create an enhanced extractor as follows:*

$\mathsf{EExt}_k(w, k||St_i) = \mathsf{Ext}(P_k(w), St_i)$ *where* $|k| = s'(k)$ *and* $|St_i| = s''(k)$.

Proof: The first thing we note about our construction is that a permutation family does not decrease the min-entropy of any distribution it acts on.

We show this as follows: Let W be a distribution where $H_\infty(W) = m$. Let P_k be a permutation family which has $|K|$ possible permutations. We now calculate the probability that after randomly selecting a permutation P_k and sampling w from W, the output is w'.

$Pr(P_k(W) = w') = \sum_{\forall k \in K} Pr(k \leftarrow K) * Pr(w \in W : P_k(w) = w')$.

For each $k \leftarrow K$ the probability that $w \leftarrow W : P_k(w) = w'$ is $\leq 2^{-m}$ by the definition of min-entropy. As we pick the key k at random, each term inside the sum is $\frac{1}{|K|}2^{-m}$. There are $|K|$ different keys k which means that there are $|K|$ different terms $\leq \frac{1}{|K|} * 2^{-m}$. Thus $\forall w', Pr(P(w) = w') \leq 2^{-m}$ which means that the over all min-entropy of the distribution gained from randomly selecting a key k and running $P_k(w)$ for sampled $w \in W$ is no less than the min-entropy of W. Thus for randomly selected St_i and for all distributions W where $H_\infty(W) = m$, (where m is the needed min-entropy of the extractor Ext_k), this construction functions as an extractor.

We now sample $w \in W$ and show that for w fixed and known to an adversary A this functions as a pseudorandom generator on $k||St_i$. Assume that there is some adversary A who can distinguish between randomly selected R_i values, and outputs of $\mathsf{Ext}(P_{k_i}(w), St_i)$ for randomly selected $k_i||St_i$. We construct A' to distinguish between the outputs of the PRP P_k for randomly selected k and random outputs.

To do this, we construct a chain of hybrid experiments H^i. Hybrid H^i is a sequence of R_j values, where the R_j values are computed as follows.

1. For all R_j values $j < i$, the R_j values are randomly selected..

2. For all R_j values $j \geq i$ the R_j values are formed by $R_j = \mathsf{EExt}(w, k_j||St_k)$ for randomly selected $k_j||St_j$.

We note that H^1 is identical to when the experiment where A is querying the function EExt and if A makes q queries the hybrid H^{q+1} is identical to the situation where A is querying the random oracle. Since A can distinguish between H^1 and H^{q+1}, by the hybrid lemma there is some i where A can distinguish between the hybrids H^i and H^{i+1}. With this in mind we construct $A'^{\mathcal{O}}$ as follows:

$A'^{\mathcal{O}}$

1. $A'^{\mathcal{O}}$ randomly selects a state St_i and queries his oracle on $\mathcal{O}(w) = w'$.
2. $A'^{\mathcal{O}}$ computes $R' = Ext(w', St_i)$.
3. $A'^{\mathcal{O}}$ gives w to A.
4. For all queries before i $A'^{\mathcal{O}}$ answers A's queries by returning randomly selected values R_j.
5. For query i $A'^{\mathcal{O}}$ returns R' to A.
6. For all queries after i, $A'^{\mathcal{O}}$ randomly selects $k_j||St_j$ and returns to A, $\mathsf{EExt}(w, k_j||St_j)$.

If $\mathcal{O}$ is the random oracle then $A'^{\mathcal{O}}$ has just created the hybrid H^{i+1} as the random oracle on w samples from the uniform distribution on $\{0,1\}^{n(k)}$. Because the uniform distribution on $\{0,1\}^{n(k)}$ has the maximum possible min-entropy, $Ext(w', St_i)$ functions as an extractor, which means that R' is ϵ close to a randomly selected string so if $\mathcal{O}$ is the random oracle $A'^{\mathcal{O}}$ gives a transcript which is statistically close to the hybrid H^{i+1} . If on the other hand the oracle is the pseudorandom permutation P_k for randomly selected k then $A'^{\mathcal{O}}$ has created the hybrid H^i. Thus, due to the fact that A can distinguish between H^i and H^{i+1} $A'^{\mathcal{O}}$ can distinguish between the pseudorandom permutation family P and the random oracle. ■

Note: For our construction, we require Ext_k to be a family of extractors where the inputs/outputs depend on k. Extractor constructions admit inputs of increasing lengths so the polynomials $n(k)$, $l(k)$, $s(k)$, $m(k)$ are explicitly constructable. Namely from the size of the output we need, $(l(k))$, and the size of the key to the pseudorandom permutation family desired, $(s'(k))$, we can construct the other parameters based on the particular extractor construction utilized.

4.2 Forward Secure Extractors

Given an enhanced extractor we can define a second tool, a forward secure extractor. This extractor is defined as follows:

A forward secure extractor FExt is a function which on input (w_i, St_i) produces a random string R_i and a new state St_{i+1}.

Definition 4.5 (Forward Secure Extractors) *A polynomial-time computable function family* $\mathsf{FExt}_k : \{0,1\}^{n(k)} \times \{0,1\}^{s(k)} \rightarrow \{0,1\}^{l(k)} \times \{0,1\}^{s(k)}$ *is an* $(n(k), s(k), m(k), l(k), \epsilon)$ *forward-secure extractor if:*

1. ***Extraction:***

   $\forall W : H_\infty(W) \geq m(k)$ $\mathsf{FExt}_k|_R$ *is an* $(n(k), s(k), m(k), l(k), \epsilon)$ *extractor.*

2. ***Forward Security***

   $\forall A, \forall W : H_\infty(W) \geq m(k)$

   $|Pr[A^{\mathcal{O}_{fr}}(w \leftarrow W) \rightarrow 1] - Pr[A^{\mathcal{O}_{fp}}(w \leftarrow W) \rightarrow 1]| \leq \alpha(k)$ *where the two oracles are defined below.*

   $\mathcal{O}_{fp}$

   - *Before any query* $\mathcal{O}_{fp}$ *samples* St_1 *randomly from* $\{0,1\}^{s(k)}$.
   - *On its i'th query, if its input is 1* $\mathcal{O}_{fp}$ *computes*

     $(R_i, St_{i+1}) \leftarrow \mathsf{FExt}_k(w, St_i)$ *and returns* R_i, *and stores* St_{i+1}.

- *If the input is 0, $\mathcal{O}_{fp}$ outputs St_i and will not respond to any further queries.*

$\mathcal{O}_{fr}$

- *Before any query $\mathcal{O}_{fr}$ samples St_1 randomly from $\{0,1\}^{s(k)}$.*
- *On its i'th query, if its input is 1 $\mathcal{O}_{fr}$ computes $(R_i, St_{i+1}) \leftarrow \mathsf{FExt}_k(w, St_i)$ and returns R_i^* randomly sampled from $\{0,1\}^{l(k)}$, and stores St_{i+1}.*
- *If the input is 0, $\mathcal{O}_{fr}$ returns St_i and will not respond to any further queries.*

We show that we can construct a forward secure extractor from any enhanced extractor. To do this, we need to utilize a tool of Bellare and Yee [10]. There they create a new tool, a forward secure pseudorandom generator. Such a generator takes a seed and outputs a random string as well as the next seed to be used. Such a generator has the property that if a seed is revealed, all strings generated from previous seeds still remain pseudorandom to the adversary. Formally:

Definition 4.6 *A FSPRG is composed of two functions,* GenFirst *and* GenNext*:*

- $\mathsf{GenFirst}(1^k) \rightarrow X_1 \in \{0,1\}^{s(k)}$
- $\mathsf{GenNext}(X_i) : X_i \in \{0,1\}^{s(k)} \rightarrow R \in \{0,1\}^{l(k)} \times X_{i+1} \in \{0,1\}^{s(k)}$

Forward Security

Consider the adversary A who has access to an oracle $\mathcal{O}$. $\mathcal{O}$ is either a pseudorandom oracle $\mathcal{O}_{gp}$, or a random oracle $\mathcal{O}_{gr}$. A can make two types of queries, a 'normal' query, or a state query. Only one state query can be made, and the adversary can make no more queries afterwards.

$\mathcal{O}_{gp}$

- *Before any queries $\mathcal{O}_{gp}$ computes $St_1 \leftarrow \mathsf{GenFirst}(1^k)$.*

- *If A queries $\mathcal{O}_{gp}$ with 1, then $\mathcal{O}_{gp}$ returns St_i.*

- *If A queries $\mathcal{O}_{gp}$ with 0, then $\mathcal{O}_{gp}$ runs $\mathsf{GenNext}(St_i) \rightarrow (R_i, St_{i+1})$. $\mathcal{O}_{gp}$ returns R_i and retains St_{i+1} for use in the next query.*

$\mathcal{O}_{gr}$

- *Before any queries $\mathcal{O}_{gr}$ computes $St_1 \leftarrow \mathsf{GenFirst}(1^k)$.*

- *If A queries $\mathcal{O}_{gr}$ with 1, then $\mathcal{O}_{gr}$ returns St_i.*

- *If A queries $\mathcal{O}_{gr}$ with 0, then $\mathcal{O}_{gr}$ runs $\mathsf{GenNext}(St_i) \rightarrow (R_i, St_{i+1})$. $\mathcal{O}_{gr}$ returns R'_i selected uniformly from $\{0,1\}^{l(k)}$, and retains St_{i+1} for use in the next query.*

We define the forward security property as follows:
$\forall A^{\mathcal{O}}\ |Pr[A^{\mathcal{O}_{gp}} \rightarrow 1] - Pr[A^{\mathcal{O}_{gr}} \rightarrow 1]| \leq \alpha$ *for negligible* α.

In [10], Bellare and Yee show that such an $FSPRG$ can be built out of a normal pseudorandom generator as follows:

Theorem 4.7 (A FSPRG Construction) *Let G be a pseudorandom generator which takes $s(k)$ bits to $l(k) + s(k)$ bits. We can construct a forward secure pseudorandom generator, (*GenFirst *and* GenNext*), as follows:*

- $\mathsf{GenFirst}(1^k)$: *Randomly sample $X \leftarrow \{0,1\}^{s(k)}$. Return X.*

- $\mathsf{GenNext}(X_i)$: *Run $G(X_i) \rightarrow r||X_{i+1}$, where $|r| = l(k)$, and $|X_{i+1}| = s(k)$. X_{i+1} is the new seed and r is the pseudorandom string.*

Proof: Full proof found in [10]. As a summary, the proof is a hybrid arguement where it is shown that if any adversary can violate the forward security of an

$FSPRG$, it can distinguish between the random oracle, and the oracle which runs the pseudorandom generator G. ■

Thanks to the above theorem, we know that forward secure pseudorandom generators exist and that they can be constructed from normal pseudorandom generators. However, a FSPRG by itself is not enough for a forward secure extractor for two reasons: it takes only one input, the random seed, while an extractor requires two; and a FSPRG can only offer computational indistinguishability of its outputs, while a forward secure extractor requires statistical indistinguishability of its outputs even given the state values. However if we use an enhanced extractor, we can gain a forward secure extractor using a similar construction as Theorem 4.7.

Theorem 4.8 (Forward Secure Extractors from Enhanced Extractors) *Let* EExt *be an enhanced extractor which takes* $\{0,1\}^{n(k)} \times \{0,1\}^{s(k)} \to \{0,1\}^{l(k)+s(k)}$ *. The following construction if a forward secure extractor.*

$\mathsf{EExt}(w, St_i) \to R_i || St_{i+1}$ *where* R_i *is the first* $l(k)$ *bits of the output of* EExt*, and* St_{i+1} *is the last* $s(k)$ *bits.*

Proof: The extraction property follows directly from the definition of an enhanced extractor.

Now, the way that we construct an FSE from an enhanced extractor is identical to the way Bellare and Yee construct a $FSPRG$ from a pseudorandom generator. Namely, we reserve the last bits of the output as the next input to be used (thinking of our FSE as a pseudorandom generator for known w which runs on input St_i). Thus by Theorem 4.7 and the definition of an enhanced extractor for known w and randomly selected initial state St_0, our construction is a forward secure pseudorandom generator.

Second, we look at the oracles $\mathcal{O}_{fp}$ and $\mathcal{O}_{gp}$ and we see that how they answer their queries and update their states is the same, with the exception that $\mathcal{O}_{fp}$ queries an FSE to generate an output and a next state, while $\mathcal{O}_{gp}$ queries a $FSPRG$. However as we have already stated, our FSE construction FExt is a $FSPRG$ for known w. Thus, $\mathcal{O}_{fp} = \mathcal{O}_{gp}$. Similarly $\mathcal{O}_{gr} = \mathcal{O}_{fr}$.

Since the two oracles are the same for our construction, for any adversary A who can distinguish between $\mathcal{O}_{fp}$ and $\mathcal{O}_{fr}$, we can construct an A' which can distinguish between $\mathcal{O}_{gr}$ and $\mathcal{O}_{gp}$ by simply echoing the queries A wishes to its own oracle, and finally echoing the answer A gives. If A' can break the $FSPRG$ based off our enhanced extractor construction for sampled but known w, then for that w our enhanced extractor is not a pseudorandom generator for that w value. So, if A can distinguish between $\mathcal{O}_{fr}$ and $\mathcal{O}_{fp}$ with non-negligible probability ϵ, A' can distinguish between $\mathcal{O}_{gr}$ and $\mathcal{O}_{gp}$ with probabiltiy $\epsilon - \nu$ for negligible ν which means that A can break the pseudorandomness of our enhanced extractor for sampled and known w with non-negligible probability.(We subtract a negligible amount from A's advantage, as we do allow a negligible probability for $\mathsf{EExt}(w, St)$ to not be a pseudorandom generator for sampled w).

■

If we are to use the enhanced extractor construction of Theorem 4.4 we will need the extractor to take a seed of size $s'(k) + s''(k)$ and output a random string of size $d(k) = l(k) + s(k) + s'(k)$. For increasing k this means that we must find biometric information that gets 'better', that is it is longer and has more entropy in order for us to gain the key size that we need. However, we note that with the enhanced extractor construction we defined earlier we can have that $s''(k)$ is logarithmic compared to $l(k)$ and that we can set $s'(k)$ to be small (but inversely polynomial) compared to $l(k)$. Now, it has been shown that there are extractors which can extractor a constant fraction of the min-entropy of their input [12]. With those we can say that the min-entropy needed for a given security parameter k is still polynomial in k as we need the min-entropy to be at least $x(l(k)+s(k)+s'(k))$ for $\exists x > 1$. As the min entropy of a distribution puts a lower bound on the size of the distribution we can also say that the needed size of our biometric is polynomial in k. We can also use less efficient extractors, as long as the extractor can extract more than a logarithmic amount of min-entropy. Moreover, some extractors may be able to provide better results if the random seed they use is larger than what is strictly necessary. Most work on extractors focuses on having very small random seeds which extract a large amount of output. Here though, we can have larger random seeds.

Next we prove a significant theorem about the states generated by the $FSPRG$ construction of Theorem 4.7. Since our construction of an FSE is a forward secure pseudorandom generator for known w, the results in this theorem apply to our FSE construction as well.

Theorem 4.9 (The states of an $FSPRG$) *Let* GenFirst *and* GenNext *be based off of the construction of Theorem 4.7. We have an adversary A who has access to the oracle O which is one of the following two oracles:*

$\mathcal{O}_{sr}$

- *$\mathcal{O}_{sr}$ begins by sampling St_0 randomly from $\{0,1\}^s$.*
- *When $\mathcal{O}_{sr}$ is queried, it returns* GenNext$(St_i) = R_i$ *and randomly selects a next state St_{i+1} from $\{0,1\}^s$.*
- *When A makes a state query, $\mathcal{O}_{sr}$ returns a randomly selected state $St_{last} \in \{0,1\}^s$.*

$\mathcal{O}_{sp}$

- *$\mathcal{O}_{sp}$ begins by running* $\mathsf{GenFirst}(1^k) = St_0$.
- *When $\mathcal{O}_{sp}$ is queried, $\mathcal{O}_{sp}$ returns $R_i = \mathsf{GenNext}(St_i)$, and stores the next state St_{i+1}.*
- *When A makes his state query, $\mathcal{O}_{sp}$ returns St_i.*

We prove that:

$\forall A$, $|Pr[A^{\mathcal{O}_{sr}} \rightarrow 1] - Pr[A^{\mathcal{O}_{sp}} \rightarrow 1]| \leq \alpha$, for negligible α.

Proof: We show that if there is an adversary A whom can distinguish between the two oracles we can construct A' which can either distinguish between the oracles $\mathcal{O}_{gp}$ and $\mathcal{O}_{gr}$, or which can break the security of the pseudorandom generator G which the $FSPRG$ is based on. A' has access to the oracle $\mathcal{O}$ which is either $\mathcal{O}_{gp}$ or $\mathcal{O}_{gr}$. A' will run A, and act as its oracle.

Such an adversary A' proceeds as follows:

- For every query made by A, A' queries $\mathcal{O}$ and receives output R_i and returns it to A
- When A makes a state query, A' makes a his state query, obtaining St and returns it to A.
- A' echoes the answer given by A. If A says it has the random oracle, A' says it has $\mathcal{O}_{gr}$. Otherwise A' says it has $\mathcal{O}_{gp}$.

First we note that under this reduction in terms of the view of A, $A'^{\mathcal{O}_{gp}} = \mathcal{O}_{sp}$ as $\mathcal{O}_{sp}$ functions exactly as a $FSPRG$, the same as $\mathcal{O}_{gp}$ and A' does not modify the queries he receives in any way.

We next need to show that $A'^{\mathcal{O}_{gr}}$ should be indistinguishable from $\mathcal{O}_{sr}$ to A. We show that if this is not the case, A is in fact capable of breaking the pseudorandom generator G. As a reminder the outputs of $\mathcal{O}_{sr}$ are G run on random state values, and a random ending state. On the other hand the outputs of $A'^{\mathcal{O}_{gr}}$ are randomly selected values and a final seed that has been updated from an initial randomly selected seed.

First, we construct a chain of hybrids E^i. Experiment E^i works as follows.

- For queries $j < i$, R_j is randomly selected.
- For queries $k \geq i$ a random state is selected and $R'_k = G(St)$ is returned. (Here, R'_k is the first l bits of $R_k = G(St)$.)
- For the final query, (the state query), a random state St is selected and returned.

For an adversary A who makes m queries, a transcript from the experiment E^m is indistinguishable from a transcript from $A'^{\mathcal{O}_{gr}}$. In both cases, the R_i values are randomly selected l bit strings. As for the states, the state returned by E^m is randomly selected, and the state returned by $A'^{\mathcal{O}_{gr}}$ is the output of a pseudorandom generator on a random input, so the states are indistinguishable as long as $G()$, the function which the forward secure pseudorandom generator is built on is a pseudorandom generator. Also we have that E^1 is identical to $\mathcal{O}_{sr}$ as each R_i value is pseudorandomly derived from a randomly selected initial state and a random final state is returned.

By the hybrid lemma and our assumption that A can distinguish between $\mathcal{O}_{sr}$ and $A'^{\mathcal{O}_{gr}}$ we know that there is some i where A can distinguish between E^i and E^{i+1}. Using this, we can construct A'' to distinguish between the random oracle, and $G()$.

A'' constructs $i-1$ random values R_i where $|R_i| = l$ and for R_i A'' queries its oracle receiving R'_i which it truncates to the first l bits. For all other values, A''

queries $G()$ on a random state and truncates the outputs to the first l bits. A'' returns these queries to A. If A'' is querying the random oracle then the value R_i' is random and thus the transcript created by A'' is identical to E^{i+1} and if A'' queried the pseudorandom oracle, then the transcript is identical to E^i. Thus, if A can distinguish between the two states with non-negligible probability ϵ, A'' can distinguish between the pseudorandom oracle G and the random oracle with probability ϵ.

So, we can assume that to A $A'^{\mathcal{O}_{gr}}$ is indistinguishable from $\mathcal{O}_{sr}$. We have already stated that $A'^{gp} = \mathcal{O}_{sp}$. Thus, if A can distinguish between $\mathcal{O}_{sr}$ and $\mathcal{O}_{sp}$ with non-negligible probability ϵ, A' should be able to distinguish between $\mathcal{O}_{gr}$ and $\mathcal{O}_{gp}$ with probability $\epsilon - \nu$ for negligible ν. (We subtract a negligible advantage as there is a negligible probability that $A'^{\mathcal{O}_{gr}}$ is not indistinguishable from $\mathcal{O}_{sr}$). ■

Having proven that the states are pseudorandom, we now prove that in the case where the biometric readings are not known, but the states are known, the extracted keys are statistically close to random. To prove that, we prove the following theorem as well.

Theorem 4.10 (The output of functions given close to random inputs) *Let $Y()$ be a function which takes l bits to n bits such that $Y(X)$, where X is the uniform distribution on $\{0,1\}^l$, outputs $x \in \{0,1\}^n$ such that $SD(Y(X), U_n) = \epsilon$. Let X' be a distribution ϵ' close to random. Then the statistical difference between $Y(X')$ and $U_n \leq \epsilon' + \epsilon$.*

Theorem 4.11 (Keys are statistically close to random) *Let* FExt *be a forward secure fuzzy extractor which produces output which is ϵ close to random where ϵ is negligible. Run* FExt, *n times, each time for a new sampled w_i value and updated state. Then the extracted keys R_i are at worst $n\epsilon$ close to random.*

Proof: First we prove Theorem 4.10.

By the definition of statistical difference:

$SD(Y(X), U_n) = \frac{1}{2}\sum_{\forall x} |Pr(Y(X) = x) - Pr(x \leftarrow U_n)| = \epsilon$. Moreover we have that:

$SD(X', X) = \epsilon' = \frac{1}{2}\sum_{\forall x} |Pr(x \leftarrow X') - Pr(x \leftarrow X)|$.

Now we calculate $SD(Y(X'), U_n)$.

$$\begin{aligned}
&= \frac{1}{2}\sum_{\forall x} |Pr(Y(X') \rightarrow x) - Pr(x \leftarrow U_n)| \\
&= \frac{1}{2}\sum_{\forall x} |Pr(Y(X') \rightarrow x) - Pr(Y(X) \rightarrow x) + Pr(Y(X) \rightarrow x) - Pr(x \leftarrow U_n)| \\
&\leq \frac{1}{2}\sum_{\forall x} |Pr(Y(X') \rightarrow x) - Pr(Y(X) \rightarrow x)| + |Pr(Y(X) \rightarrow x) - Pr(x \leftarrow U_n)| \\
&= \frac{1}{2}\sum_{\forall x} |Pr(Y(X') \rightarrow x) - Pr(Y(X) \rightarrow x)| + \epsilon \\
&= \frac{1}{2}\sum_{\forall x} |\sum_{\forall a} Pr(a \leftarrow X')Pr(Y(a) \rightarrow x) - Pr(a \leftarrow X)Pr(Y(a) \rightarrow x)| + \epsilon \\
&= \frac{1}{2}\sum_{\forall x} |\sum_{\forall a}(Pr(a \leftarrow X') - Pr(a \leftarrow X))Pr(Y(a) \rightarrow x)| + \epsilon
\end{aligned}$$

Now, we note two things. First, for each a and x, $Pr(Y(a) \rightarrow x)$ is either 1 or 0. Moreover, if $Pr(Y(a) \rightarrow x)$ is 1 for some a it is 1 for one and only one x. $Pr(Y(a) \rightarrow x') = 0$ if $Pr(Y(a) \rightarrow x) = 1$ for some $x \neq x'$. Thus for each a, the term $Pr(a \leftarrow X') - Pr(a \leftarrow X)$ will appear once and once only over all the x values in the outer sum. Thus we can drop the outer sum over all x and deal only with the sum over all a.

$$
\begin{aligned}
&= \frac{1}{2} \Big| \sum_{\forall a} Pr(a \leftarrow X) - Pr(a \leftarrow X') \Big| + \epsilon \\
&\leq \frac{1}{2} \sum_{\forall a} |Pr(a \leftarrow X) - Pr(a \leftarrow X')| + \epsilon \\
&= \epsilon' + \epsilon
\end{aligned}
$$

Thus we have that $SD(Y(X'), U_n) \leq \epsilon + \epsilon'$.

Going on, to apply Theorem 4.10 to Theorem 4.11 we simply need to note that for a w sampled from W the function $\mathsf{FExt}(w, St_i)$ is a function that outputs a string that is ϵ close to random. Also, the first time we utilize the forward secure extractor the key St_0 is random. Thus R_0, the full output of the extractor after the first time it is run is ϵ close to random which means that St_1, the last $s(k)$ bits is ϵ close to random. Going on we can use Theorem 4.10 to say that for the next use of the extractor the key is 2ϵ close to random, for the third use it is 3ϵ close to random and so on. Thus for n uses, the extracted keys are $n\epsilon$ close to random. For polynomial n and negligible ϵ this means that all keys are negligibly close to random. This holds even if the states are known, due to the definition of an extractor. ■

Having now described and constructed a forward secure extractor, we show how an enhanced forward secure fuzzy extractor, $(EFSFE)$, can be constructed.

4.3 Enhanced Forward Secure Fuzzy Extractor Construction

With possession of one construction for a forward secure extractor we construct a forward secure fuzzy extractor. We show that given our previous forward secure extractor construction and a fuzzy sketch construction, we can construct a forward secure fuzzy extractor construction. As our previous forward secure extractor construction only required the existence of an enhanced extractor, this means that an enhanced forward secure fuzzy extractor can be built out of an enhanced extractor, a fuzzy sketch, and a pseudorandom function F.

Theorem 4.12 (An Enhanced Forward Secure Fuzzy Extractor) *Given the forward secure extractor* $\mathsf{FExt}(w', St_i)$ *constructed earlier, a fuzzy sketch* $(\mathsf{Fsk}, \mathsf{Rec})$ *and a pseudorandom function* F *the following is an enhanced forward secure fuzzy extractor.*

- $\mathsf{FirstKey}(1^k)$*: Randomly sample* $St_0 \leftarrow \{0,1\}^{s(k)}$*.*
- $\mathsf{Rep}(w', St_i, P_i)$*: Run* $\mathsf{Rec}(w', P_i)$ *gaining* w_i*. Run* $(R_i, St_{i+1}) = \mathsf{FExt}(w_i, St_i)$ *and output* R_i*.*
- $\mathsf{Update}(w', St_i, P_i)$*: Run* $\mathsf{Rec}(w', P_i)$ *gaining* w_i*. Run* $\mathsf{FExt}(w_i, St_i) = (R_i, St_{i+1})$ *and output* St_{i+1}*.*
- $\mathsf{EUpdate}(w^*, St_i)$*:* $St_{i+1} = F(w'^* \oplus St_i)$*, where* w'^* *is* w^* *truncated so that* $|w'^*| = |St_i|$*. Return* St_{i+1}*.*
- $\mathsf{Gen}(w', St_i)$*: Run* $\mathsf{Fsk}(w')$ *forming a public sketch* P_i*.*

Proof: The correctness of the fuzzy extractor is the simplest part of this proof. Let St_i be any state information. We have $d(w, w') \leq t$, where t is the error tolerance of Rec. This means that if $P' = \mathsf{Gen}(w', St_i) = \mathsf{Fsk}(w')$, then $Rec(w', P') = Rec(w, P') = w'$, and for identical seeds St_i and St'_i, $\mathsf{FExt}(w_i, St_i) = \mathsf{FExt}(w_i, St'_i)$. Putting these two equalities together for the function Rep, we get that:

$\mathsf{Rep}(w, St_i, P') = \mathsf{FExt}(Rec(w, P'), St_i) = R = R' = \mathsf{FExt}(Rec(w', P'), St_i) = \mathsf{Rep}(w', St_i, P')$.

The extraction property is similar. For any state St, FExt is an extractor by definition and Gen is a public sketch. We do not publish the state St in the public sketch, however if we did our construction of Gen and Rep would be identical to the construction in Theorem 2.6. Moreover, the only purpose in publishing the state in the public sketch is so that it can be retrieved and used again. We simply do privately what could be doine publicy and still have the extraction property. Thus, from Theorem 2.6 we can gain the extraction property that $SD((R, P), (U_l, P)) \leq \epsilon)$.

Now we go and prove the forward security property of this construction.

Forward Security

We divide the proof up into the two cases. First, we assume that the adversary A makes both a biometric corruption and state corruption query. Moreover, we assume that this biometric corruption query allows him to know all biometric readings w_i. Later, we assume he makes only one type of corruption query.

We assume that there is an adversary A whom can violate the forward security of our $FSFE$ after he makes both types of corruption queries. We construct A' which can break the forward security of the underlying FSE, FExt. A' works as follows:

1. A' will first receive w and then will make one normal query receiving R and one state query receiving St'.

2. A' will pick an initial state, $St_0 = \mathsf{FirstKey}(1^k)$.

3. A' will guess a query $d \in GS$, the decide query made by the adversary. If the adversary does not make the decide query at d, A' will randomly flip a coin.

4. A' selects a distribution set W where $w \in W$ and the min-entropy of W is high enough.

5. For all queries $i < d$ made by A, A' will answer in the following way:

   - For each query A' will sample $w_i \in W$, and publish $P_i = \mathsf{Gen}(w_i, St_i)$
   - After each query A' will either update using Update or $\mathsf{EUpdate}$ at the behest of A, selecting new w^* as appropriate.
   - For an insider query A' returns $\mathsf{Rep}(w_i, St_i, P_i) = R_i$.

6. For query d A' will set $P_i = \mathsf{Gen}(w, St_r)$ and $R_d = R$.

7. For query $d+1$ A' will set $St_{d+1} = St'$.

8. A' will continue to answer queries as long as A desires them.

9. When A responds with his guess query, A' will echo A's guess, i.e. A' will say pseudorandom if A says pseudorandom and vice versa.

We assume for the moment that A' selected the correct query. At the end, we can simply divide the advantage of A' in this case by *total* where *total* is the number of queries. Where *total* is polynomial in $|St|$, then the resulting advantage is still polynomial. We prove that this reduction is successful by showing that the action of A'^{O} is indistinguishable from the action of $\mathcal{O}_{EFE}$ to A if the oracle queried by A' is $\mathcal{O}_{fr}$ we have that $\mathcal{O}_{EFE}$, (to A), returns a random value for A's decide query and if the oracle queried by A' is $\mathcal{O}_{fp}$ then to A, O_{EFE} returns a valid extraction.

We note that $A'^{\mathcal{O}}$ functions exactly like an enhanced forward secure fuzzy extractor for all queries before query d. It samples from a distribution set with high enough entropy, selects a valid initial state and runs functions as appropriate. As A' acts like a $EFSFE$ for all queries before query d, A's view on those queries is indistinguishable from $\mathcal{O}_{EFE}$. We can say the same thing for all queries $q > d+1$. We now focus on query $d+1$ and query d, the decide query, and we show that in these two cases as long as $d \in GS$, (which we hold as true by our assumption), for these two queries the action of $A'^{\mathcal{O}}$ is actually indistinguishable from the action of $\mathcal{O}_{EFE}$.

We look at each element of the queries in turn. First, we can say that in both queries the public sketch A gives is a valid public sketch of the biometric as A' has knowledge of the biometric to be used in each turn and creates valid sketches. Also, the sketch function Gen does not utilize the state so the state St_r is not used in the sketch and thus remains invisible. Also, the biometric string used for query $d+1$ is validly sampled, and the distribution W contains w so the biometric strings come from a distribution with high enough entropy.(A' cannot tell the difference between sampling a biometric for query d and returning w as A' has no idea of the distribution it is dealing with, or the distribution A''s oracle is sampling from). For query d, either the string R is a random string unrelated to the biometric sketch and state ,(if A''s oracle is $\mathcal{O}_{fr}$), or the string R is the extraction of the biometric w using state St''. Also, the string R_{d+1} is a valid extraction of w_{d+1} using state St'. So the string $R = R_d$ is exactly one of the possible strings $\mathcal{O}_{EFE}$ could create, as it is either a random state independent from all other information, or a valid extraction on the biometric w for some state. The difference between $A'^{\mathcal{O}}$, and $\mathcal{O}_{EFE}$ are the states used for query d, and for query $d+1$. The state St_d is not updated from the state St_{d-1} as it should be. Next, we show that if $d \in GS$ these states are still indistinguishable from the 'true' states.

Case 1:

In this case, $d < s_1$. Here, we can use Theorem 4.9. The theorem states that for a FSE, the updated states are computationally indistinguishable from randomly selected states for all states before the adversary makes a state query (which is identical to a state corruption query). Thus the state St_d, a randomly selected state, (as it is the first state the oracle O generates), is indistinguishable from the state $St_{d_{true}}$, the state that comes from $Update(w_{d-1}, St_{d-1}, P_{d-1})$. Also we see that the state St' is validly updated from St_d, no matter whether or not A' has access to the oracle $\mathcal{O}_{fr}$ or $\mathcal{O}_{fp}$. Thus in this case the states are computationally indistinguishable from the 'true' states A would expect from $\mathcal{O}_{EFE}$.

Case 2:

In this case $d > e_i$ and there is no state corruption query s_j where $d > s_j > e_i$. Here we show that A's knowledge of the queries $e_i < q < s_j$,(where s_j is the next state corruption query after e_i), is the same as A's knowledge of the queries in Case 1. To do this, we show that the state St_{e_i+1} is computationally indistinguishable from random even given all previous information. We show that if this is not the case then we can break the pseudorandom function F.

The reduction is rather simple. We denote A as the adversary capable of distinguishing between a random state St_{e_i+1} and $St_{e_i+1} = \mathsf{EUpdate}(w^*, St_{e_i}) = F(w^{**} \oplus St_i)$, (where w^{**} is w^* truncated so that $|w^{**}| = |St_i|$), given previous information, and we construct A' which will break the pseudorandom function F. The reduction is similar to the larger reduction that we are in presently. Namely:

- $A'^{\mathcal{O}}$ guesses when A will make the query e_i. (Again we assume for the sake of the proof that A' guesses right, knowing that we divide A''s advantage by $total$).
- For all queries before e_i, A' simulates a forward secure fuzzy extractor.
- For the enhanced update query e_i, A' updates using $St_{e_i+1} = \mathcal{O}(w^{**} \oplus St_{e_i})$.
- For all later queries A' simulates a $FSFE$.

It is plain that if A' guesses correctly and $\mathcal{O}$ is the random oracle, then the state St_{e_i+1} is random, otherwise the oracle St_{e_i+1} is a pseudorandom function of $w^{**} \oplus St_{e_i}$. Thus, if A can distinguish between a randomly selected state St_{e_i+1} and a pseudorandomly selected state with probability $\geq \epsilon$ for ϵ non-negligible, A' can break the pseudorandom function F with probability $\frac{\epsilon}{total}$. As long as $total$ is polynomial in the security parameter A''s advantage is still non-negligible.

Because the state St_{e_i+1} is indistinguishable from a random state, even to an adversary who knows every piece of information in all previous queries, we can say that A's view on queries $e_i + 1$ thru s_j is indistinguishable from A's view on queries 1 thru s_1. The first query in both cases is with a sampled biometric and random unknown state St. Each subsquent query is a query with sketch, sampled biometric, and updated state. The final query in both queries is a state query. For all queries after the enhanced update, A has the capability to learn the same things that he could learn at the first query. Namely, he can learn the biometric string in every round, he can make as many insider queries as he pleases, and he views all public strings in each case automatically. As the only thing that remains secret is the states, and those are indistinguishable from random states selected independently of all previous states, all previous queries do not allow the adversary to learn anything about the states after query e_i and so we can say that the knowledge A has about the queries between e_i and s_j is the same as the knowledge A has about the queries 1 thru s_j.

Since the queries between e_i and s_j are indistinguishable to the queries 1 thru s_1 in terms of an adversaries knowledge of the states, public strings, keys and biometric information, we can say that if $e_i < d < s_j$, then the analysis we developed in Case 1 still applies, and thus the queries d and $d+1$ are indistinguishable from $\mathcal{O}_{EFE}$ to A.

Having shown that the action of $A'^{\mathcal{O}}$ is indistinguishable from the action of $\mathcal{O}_{EFE}$, A' gains part of A's advantage. Thus, if A's guess query is correct with probability ϵ, then A' will distinguish between $\mathcal{O}_{fr}$ and $\mathcal{O}_{fp}$ with probability $\frac{\epsilon}{total}$, and if *total* is polynomial in the security parameter, the probability is non-negligible.

Now we assume that the adversary A makes only one type of corruption query, either a biometric corruption query or a state corruption query.

First, we deal with the case where an adversary does not ever make a state corruption query. Here we say that the set GS encompasses all the queries made by the adversary, and thus all keys in that set should remain secure. If the adversary makes m queries, then the first state corruption query he could ever make would be a query $m+1$, which would mean that all m queries are in GS. Thus, by the analysis in the previous case, we can say that all keys remain indistinguishable from random.

Next, we assume that the adversary makes some state corruption queries, but no biometric corruption queries. We also assume that the adversary learns all possible states St_i. However, since the adversary makes no biometic corruption queries, we can use Theorem 4.11 and say that all the keys R_i are negligibly close to random which means that all adversaries have at best a negligible chance of success in distinguishing a random key from the true extracted key.

■

4.4 Robustness in an Enhanced Forward Secure Fuzzy Extractor

Here we show how to add the robustness property to our enhanced forward secure fuzzy extractor construction. To do this we will utilize the method Dodis et al. used in [7]. There they developed a sketch protocol that contained authentication information, such that any adversary who does not know the key cannot forge this information. We use the same idea here. First, we introduce the following tool.

Definition 4.13 (MAC) *A MAC is any function, $MAC_{St}\{0,1\}^n \to \{0,1\}^v$, which has the following two properties.*

- *For all x and δ, $Pr_{St}[MAC_{St}(x) = \delta] = 2^{-v}$.*
- *For all $x \neq x'$ and for any δ, δ', $Pr_{St}[MAC_{St}(x') = \delta' | MAC_{St}(x) = \delta] = \alpha(v)$, for negligible α.*

An extractor MAC is a MAC that is also an extractor, where *St* is the seed to the extractor. While a regular MAC hides the key computationally given the message and MAC, an extractor MAC hides the key statistically as long as the message has high enough entropy to the adversary. Moreover, for a MAC it should be difficult for any adversary, given multiple different messages and tags under a key to output a new message and a valid tag.[2] In [7], explicit constructions of an extractor-MAC are given utilizing pairwise independent hash functions.

Using an extractor-MAC function MAC, we alter the function Gen of our enhanced forward secure fuzzy extractor as follows.

$\mathsf{Gen}'(w_i, St_i) = (P'_i = \mathsf{Fsk}(w_i), \pi = MAC_{St_i}(w_i \oplus St^*_i, P'_i))$. (Here St^* is St concatenated to itself enough times such that $|w_i| = |St^*|$.)

With this function in place of the old function Gen we construct a robust enhanced forward secure fuzzy extractor as follows:

Theorem 4.14 (A Robust Enhanced Forward Secure Fuzzy Extractor) *The following is an $REFSFE$.*

- $\mathsf{FirstKey}(1^k)$: *Randomly sample* $St_0 \leftarrow \{0,1\}^{s(k)}$.
- $\mathsf{Rep}(w', St_i, P_i = (P'_i, \pi))$: *Run* $Rec(w', P'_i)$ *gaining* w_i. *Check to see if* $\pi = MAC_{St_i}(w_i \oplus St^*_i, P'_i)$. *If not, output* $\perp$, *otherwise run* $R_i, St_{i+1} = \mathsf{FExt}(w_i, St_i)$ *and output* R_i.
- $\mathsf{Update}(w', St_i, P_i)$: *Run* $Rec(w', P'_i)$ *gaining* w_i. *Run* $\mathsf{FExt}(w_i, St_i) = R_i, St_{i+1}$ *and output* St_{i+1}.
- $\mathsf{EUpdate}(w^*, St_i)$: $St_{i+1} = F(w'^* \oplus St_i)$, *where* w'^* *is* w^* *truncated so that* $|w'^*| = |St_i|$. *Return* St_{i+1}.
- $\mathsf{Gen}(w_i, St_i) = \mathsf{Gen}'(w_i, St_i)$.

To prove this construction is robust, we prove the following theorem. We show that for any adversary who does not know the state St_i , the probability of outputting a public string $P'_i \neq P_i$ that breaks robustness is negligible.

Theorem 4.15 (Unforgeability) *$\forall$ adversaries A, where $\exists i$ such that A does not know St_i, $Pr[A(Tr) \rightarrow P_i^* : P_i^* \neq P_i$ and $\mathsf{Rep}(w_i, St_i, P_i^*) \neq \perp] \leq \alpha$, for negligible α, where Tr is a transcript of A's interaction with the oracle $\mathcal{O}_{rob}$.*

Proof: There are two cases. The first case is when A outputs $P_i^* = (P_i'^*, \pi^*)$ where $P_i'^* = P_i'$. In this case π^* must be $MAC_{St_i}(w_i \oplus St_i^*, P_i' = P_i'^*)$ for A to succeed as $\mathsf{Rec}(w_i, P_i'^*) = w_i$. However, that means that $\pi^* = \pi$ which means that $P_i^* = P_i$ and A has not violated robustness.

In the second case, $\pi^* = \pi$. Here, $P_i'^* \neq P_i'$ and A should not be able to create a tag π^* such that $\pi^* = MAC_{St_i}(\mathsf{Rec}(w', P_i'^*) \oplus St_i^*, P_i'^*) = \pi = MAC_{St_i}(\mathsf{Rec}(w', P_i') \oplus St_i^*, P_i')$ as $(Rec(w', P_i'^*) \oplus St_i^*, P_i'^*) \neq (Rec(w', P_i') \oplus St_i^*)$due to the second property of a MAC.

So, with all but negligible probability A cannot create a valid tag for the MAC without knowing the state. ■

With the above theorem proven robustness follows simply as the MAC hides the key, (at least computationally), and the random values R should not reveal anything about the states St_i as the strings should be random even given the St_i information. Moreover, we gain robustness even when the adversary is allowed to know the biometric inforation w_i for a query.

Chapter 5

Uses of Forward Secure Fuzzy Extractors

In the previous chapter we introduced a construction for an enhanced forward secure fuzzy extractor. This extractor retains the security of its keys even in regards to the theft of biometric information. In this chapter we examine several different authentication schemes that utilize fuzzy extractors and examine how they can be changed to allow for the use of a forward secure fuzzy extractor. We also construct protocols that allow two individuals to utilize an enhanced forward secure fuzzy extractor and use the enhanced update function, even when one individual does not have access to the biometric data source of the other.

5.1 The Use of Fuzzy-Extractors in Authentication Schemes

By themselves, fuzzy extractors allow for a strong cryptographic key to be utilized easily and without the key being electronically stored. It is not immediately clear how this key may be used in different protocols. This section explores the different ways in which a fuzzy extractor can be utilized in an authentication protocol. These protocols mainly differ in what information the server is allowed to possess. In all these protocols, it is assumed that the user stores no information except for the second factor, the state information. The first two come from [4] and [13] while the last is, as far as we know, original to this paper.

Theorem 5.1 (An Asymmetric Authentication Protocol) *The following protocol is an unidirectional authentication scheme.*

- *User computes* $(R, P) = Gen(w)$*, and computes* PK_R*, the verification key of a signature scheme where* R *is the signing key. If* R *is not a valid signature key, one is deterministically developed from* R*. These keys are certified by a trusted certification authority. The user passes* PK_R*,* P *to the server.*
- *For authentication the user inputs a new instance of the biometric data* w'*.*
- *The server sends* P *and a random nonce* n *to the user.*
- *Using* P*, the user computes* $R = Rep(w', P)$*, and signs a message* $m||n$ *using* R*.*
- *The server verifies the signature using the verification key* PK_R*.*

Proof: Full proof is found in [4]. Informally, without access to the signing key R, no adversary should be able to sign $m||n$, and R is ϵ close to random even given P due to the fuzzy extractor. ■

This construction has the advantage that the only information that is stored in the server is information that can be made public, so the security of the server's data is not an issue. Also, there is an implicit idea of robustness here. Namely, after signing the message $m||n$, the user can verify that the signature is correct by using the verification key, which is public. However, this is a one way authentication protocol in that the server cannot authenticate himself to the verifier under this scheme. Moreover, for forward secure fuzzy extractors this protocol is only good for one authentication, in that the updating of the state St, will result in a new secret key R, and will require the user to send to the server a new public key, $PK_{R_{i+1}}$ which would require this key to be certified as well. This is necessary as the server cannot update the state itself. Next we present one that acheives mutual authentication.

Theorem 5.2 (A Mutual Authentication Protocol) *The following protocol is a mutual authentication scheme.*

- *Let π be a Password Authenticated Key exchange protocol.*
- *User computes $(R, P) = Gen(w)$, and sends R and P to the server.*
- *For authentication, the user receives P from the server, and computes R.*
- *The server and the user both run π for authentication, using the key R.*

Proof: Full proof is found in [13]. Informally the fuzzy extractor allows both the server and the user to arrive at the same key R. The security of the protocol depends on the security of the PAK protocol π and the security of the fuzzy extractor. ■

Here, we do have mutual authentication at the cost of having the server store sensitive information. Again, to update to the next state in this protocol, after the user and the server are mutually authenticated and they have a session key developed by the key exchange protocol, the user computes the new state, extracts using the same public string and the new state and sends the key over to the server. Unfortunately, given the lack of w, and state information St_i, the server cannot update from state to state itself, either deterministically or through an enhanced update. This means that while the alteration to this protocol is simple, it does create large overhead as each authentication must end in sending the state for the next authentication.

Another possible mutual authentication scheme would be if instead of the server storing R, the server stores an instance of the biometric data w'. Now authentication takes place when the user runs $\mathsf{Gen}(w)$, creating R, P and sends P to the server. If $d(w, w') \leq t$ then both the user and the server will arrive at the same key, R, which can be used in a PAK protocol. Moreover even if we do not use a forward secure fuzzy extractor for this protocol, we have the property that different keys can be developed for each authentication as different public strings P will have the user and server agree on different biometrics which will (may) extract to different values. Also, if the server stores the state information as well, this allows the server and the user to update independently, something the previous two protocols have lacked. However, it has an unfortunate drawback in that the server is now storing the user's biometric data which can lead to a far more damaging breach in security if the server's data is compromised or if the server maliciously uses the data, than if the server simply stored information specific to the authentication protocol. Knowledge of a user's biometric information can allow a malicious server or adversary to corrupt other protocols where the user used the same biometric source of information. However, if we allow the user and server to share a pseudorandom function, we can eliminate this drawback with the following protocol.

Note: For this protocol, it is necessary that the 'fuzzy' data comes from a metric space $\mathcal{M}$, which has the property that there is a family of isometries $I(w, s)$ that $\forall s, s'$ has the property that $I(w, s) - s + s' = I(w, s')$. The space of bit strings of fixed length, under the Hamming Metric is one such space, where $I(w, s) = w \oplus s$, and the operation $-s + s' = \oplus s \oplus s'$.

Theorem 5.3 (A Mutual Authentication Scheme) *Let π be a PAK protocol. Let F be a pseudorandom function. Let* Gen *and* Rep *be our forward secure fuzzy extractor, where* Rep *has error tolerance t. The following is a mutual authentication protocol.*

- *The user computes his beginning state, $FirstKey(1^k) = St_0^u$. He also computes the joint first state $FirstKey(1^k) = St_0$ independently.*
- *The user computes $w^* = I(w, F(St_0^u))$, and sends St_0, w^* to the server. He stores St_0^u.*
- *To authenticate, the user computes $P' = Gen(w'^* = I(w', F(St_i^u)), St_i)$ and computes $R_u = Rep(w'^*, St_i, P')$ and sends P' to the server.*
- *The server computes $R_s = Rep(w^*, St_i, P')$.*
- *The user and server authenticate themselves by running the PAK protocol π with keys R_s, R_u.*
- *If authentication is successful, the user computes $St_{i+1}^u = F(St_i^u)$, and sends $X = -F(St_i^u) + F(St_{i+1}^u)$ to the server. The server takes w_i^* and stores $w_{i+1}^* = w_i^* + X$.*
- *Both the user and the server update using P', St_i and $Rec(w'^*, P')$ and $Rec(w^*, P')$ respectively.*

Proof: If after both the sever and user have extracted keys based off the information they have, and these keys are equal then they should mutually and securely authenticate themselves based off of the PAK protocol. If $d(w, w') \leq t$ where t is the error tolerance of the fuzzy extractor, then:

$d(I(w', F(St_i^u)), I(w, F(St_i^u))) \leq t$

by the property of the isometry I. So, as long as $d(w', w) \leq t$, both the server and user should agree on $I(w', F(St_0^u))$ as the data the the fuzzy sketch is storing. Due to the deterministic updates of the state St_i and for each update both the user and server should agree on the biometric input to use to update, then both the server and user should agree on the same state St_{i+1}, which means that they should both be able to extract the same key for all states, which will enable the PAK protocol to function after all updates.

For an adversary trying to authenticate himself in this protocol, the first thing we note is that the probability that he picks the right state is negligible, as the states are pseudorandom, even given the public sketches P_i. If an adversary, given the public strings P_i and even given some keys R_i should be able to select a state, then he can break the forward secure fuzzy extractor. Moreover, even if the adversary were to guess the right state St_i, he still must create sketch that is 'close' enough either to the original biometric reading w^* and also know the state St_i^u or publish a sketch that is close enough to a randomly shifted element $w^* \in \mathcal{M}$. The chances of him performing either attack is negligible, even given the error correcting abilities of the fuzzy sketch.

Also, this protocol hides a user's biometric information in the sight of the server, in that the information the server sees is not the true biometric string w rather a shifted string, $Y_i = I(w, F(St_i^u))$, for pseudorandom St_i. If the server A can distinguish Y_i from a sequence of random bits, we can construct A' to break the pseudorandom function $F()$ as follows.

- A' selects a random beginning string St_0 and samples $w \leftarrow W$.
- For $i = 1, 2, ...n$.
- A' queries his oracle $\mathcal{O}(S_i)$, and gains output S_{i+1}. He creates the string $w_i^* = I(w, S_i)$, which he sends to A.

If the oracle for A' is a pseudorandom function, then the distribution on strings w_i^* match exactly what A would receive if A received information from the user, namely biometric strings $I(w, F(St_i^u)$ for updated states St_i^u. Thus, when $\mathcal{O}$ is the pseudorandom function we gain the advantage that the adversary A possess. If that advantage is non-negligible, then the advantage given to A' is non-negligible. However, note that if an adversary eavesdrops on the strings $X_i = -F(S_i^u) + F(S_{i+1}^u)$, from $i = 1, 2, ..., j$, and then receives w_{j+1}^* he can invert w_{j+1}^* back to w_0^*, as the server's biometric update operation $w_{i+1}^* = w_i^* + X$ is easily invertible. Do we still have that the strings $I(w, S_i^u)$ are in fact pseudorandom even given the strings X_i? The reduction to prove this is similar. Now, an adversary A' not only creates strings W_i^*, but using the S_i values he receives from his oracle he creates $-S_i + S_{i+1}$ and sends those to A if A desires to see a specific X_i string. ■

In this last authentication protcol an interesting thing that we did was reverse how the protocol was done, in that in previous protocols we had the server store the fuzzy sketch, and the user request it while in the last authentication protocol we had the user created a new fuzzy sketch each time. Here, we do not gain authentication in that the 'correct' key is created, as we create a different key each time. We gain authentication because we have that the server and the user will always agree on the same key, while for anyone who does not know the biometric, unless he is lucky enough and picks one close enough to the shifted biometric reading stored by the server and selects the correct second factor, he can not gain access. This is rather different than other authentication models where the key remains fixed. Moreover, for an adversary who does not learn any of the server's or user's private information, if he was capable of finding the key R used in one execution of the PAK protocol, that would not help him find other keys as the next key utilized would be independently selected due to the fact that the keys are based off the biometrics selected for each round.

5.2 The Enhanced Update Procedure

In the previous authentication protocols, the enchanced update protocol cannot be performed by the server independently of the user. If there was any way for the server to perform the enhanced update $\mathsf{EUpdate}(w', St_i)$, without knowledge of w', or with only approximate knowledge of w', then any adversary should be able to perform the enhanced update himself. Here we show two possible methods which minimize the cost of the fact that the server cannot perform the enhanced update himlself.

1. We can 'amortize' the cost of a fuzzy update.

2. The bounded storage model. In this model, the fuzzy update is not performed utilizing the user's biometic source W, but rather it utilizes an extremely large random string LR that the user and server have access to. The security of the update relies on the fact that the adversary is limited in storage.

5.2.1 Amortize the Cost

This idea is relatively simple. We simply accept the idea that the enhanced update can only be performed occaisonally. An example in terms of authentication protocols is that after the user and the server have authenticated to each other, and have agreed on a session key, the user performs an enhanced update, encrypts the new state under the session key, and sends it to the server. This allows the enhanced update to be performed after every authentication step with minimal overhead, however should a forward secure fuzzy extractor be used as part of another protocol this is not assured.

5.2.2 Bounded Storage Model

The bounded storage model was first introduced by Maurer. The only constraints placed on the adversary in this model is that the adversary cannot store more than m bits of information for some variable m. The user and the server has access to a string that is of size $|K| > m$ bits. This model can be thought of as the case where the user and the server have access to a streaming source of randomness with high channel capacity. This means that only a small amount of the string can be stored by any adversary A. Cachin and Maurer in [6], develop a key generation scheme, (which we call $BSKG$), where given access to the source of randomness RS two people can agree on a key in such a way that any adversary with bounded storage will only have a negligible knowledge of the key. Moreover, this is a keyless approach, in that the user and the server need not share a key to create a new key. With this in hand, performing a fuzzy update in the bounded storage model is relatively simple. Through $BSKG$ in [6] have the server and user create a key w_k where $|w_k| = |St_i|$. You then have $St_{i+1} = F(St_i \oplus w_k)$, for pseudorandom F, and the analysis done on the enhanced update procedure still holds.

Given the parameters of the bounded storage model, it is infeasible for it to be performed quite frequently as to perform the enhanced update in this model both the user and the server must sample from a very large string, (terabytes or larger). However, if we perform the enhanced update in a protocol such as the one described in Theorem 5.3 then between updates we can still arrive at different keys used for the PAK protocol which is nice as if the adversary gains one such key for a previous execution of the PAK protocol the others remain secure.

Chapter 6

Conclusion

In this thesis we have examined the use of biometric information in creating cryptographically strong keys in a method that still offers security even when the biometric information is maliciously recovered at some point. To do this, we have defined the following primitives.

1. An enhanced extractor, an extractor that becomes a pseudorandom generator when the low entropy string is revealed but the state remains unknown.

2. A forward secure extractor, an extractor which updates its second factor in such a way that if this factor if both factors are known at a specific state, all keys generated before the adversary learned the second factor remain secure.

3. A forward secure fuzzy extractor, a stateful fuzzy extractor where all keys before an adversary retrieves some state information remain secure, even if the biometric information is revealed at some point, and where some keys still remain secure even if both factors are maliciously recovered.

We have shown that given an enhanced extractor, we can construct a forward secure fuzzy extractor. In addition we have thoroughly explored the possible security properties of such a fuzzy extractor. We developed an enhanced update function which allows for keys developed after the enhanced update function to be secure. We also defined the idea of 'strong' forward security, where if the adversary only retrieves one of the two factors, all extracted keys remain secure, and we have shown how this can be acccomplished using previously constructed tools. We have also defined the idea of how we can obtain robustness in a forward secure fuzzy extractor, even when an adversary can steal the biometric information involved which is a stronger property than previous constructions.

Our construction, being a two factor construction offers greater security properties than previous biometric only constructions, at the cost of storing the second factor. It is also important to note that our enhanced security properties are always in terms of the size of the second factor of our construction. This means that for increasing levels of security we will need to utilize increasingly 'good' biometric information, that is, information that has more and more entropy and gets larger and larger in size. The precise parameters which describe the needed increase in entropy will depend on the enhanced extractor construction utilized.

Bibliography

[1] A. ADLER. Images can be regenerated from quantized match score data. In *Canadian Conference on Electrical and Computer Engineering, Vol 1.*, pages 469–472, 2004.

[2] KILLIAN BELLARE AND PHILLIP ROGAWAY. The security of the cipher block chaining message authentication code. In *Journal of Computer and System Sciences 61*, pages 362–399, 2000.

[3] MIHIR BELLARE AND PHILLIP ROGAWAY. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

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

[5] XAVIER BOYEN, HOVAV SHACHAM, EMILY SHEN, AND BRENT WATERS. Forward-secure signatures with untrusted update. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 191–200, New York, NY, USA, 2006. ACM Press.

[6] CHRISTIAN CACHIN AND UELI MAURER. Unconditional security against memory-bounded adversaries. In *Advances in Cryptology — CRYPTO '97*, Burton S. Kaliski Jr., editor, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer-Verlag, August 1997.

[7] YEVGENIY DODIS, JONATHAN KATZ, LEONID REYZIN, AND ADAM SMITH. Robust fuzzy extractors and authenticated key agreement from close secrets. In *CRYPTO*, pages 232–250, 2006.

[8] YEVGENIY DODIS AND ADAM SMITH. Correcting errors without leaking partial information. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 654–663, New York, NY, USA, 2005. ACM Press.

[9] Marcos Faundez-Zanuy. On the vulnerability of biometric security systems. In *Aerospace and Electronic Systems Magazine, IEEE*, pages 3 – 8, 2004.

[10] Bennet Yee Mihir Bellare. Forward security in private-key cryptography. In *Topics in Cryptology CT-RSA-2003*, pages 1 – 18. Springer Berlin/ Heidelberg, 2003.

[11] Jonathan Katz Ran Canetti, Shai Halevi. A forward secure public key encryption scheme. In *Advances in Cryptology - EUROCRYPT '03*, pages 255 – 271. Springer Berlin, 2003.

[12] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. In *STOC '99: Proceedings of the thirty first-annual ACM symposium on Theory of computing*, New York, NY, USA, 1999. ACM Press.

[13] Jonathan Katz Rafail Ostrovsky Xavier Boyen, Yevgeniy Dodis and Adam Smith. Secure remote authentication using biometric data. In *Advances in Cryptology, Eurocrypt 2005*, pages 147–163. Springer Berlin / Heidelberg, 2005.

[14] Adam Smith Yevgeniy Dodis, Leonid Reyzin. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology, Eurocrypt 2004*, pages 523–540. Springer Berlin / Heidelberg, 2004.

VITA

David Goldenberg

David Goldenberg was born August 25, 1983. He went to high school at W.T. Woodson in Fairfax VA. Graduating high school he attended The College of William and Mary for his undergraduate degree, obtaining a B.A. in theatre with a minor in mathematics. After graduation he went straight into the graduate computer science program at William and Mary. David is interested in many things in computer science including, but not limited to: AI, Statistical learning, Video Games, Cryptography, Information Theory, and Simulations.