SelfIs: Self-Sovereign Biometric IDs

Luis Bathen, German H. Flores, Gabor Madl, Divyesh Jadav
IBM Research
{bathen,ghflores,madl,divyesh}@us.ibm.com

Andreas Arvanitis, Krishna Santhanam, Connie Zeng, Alan Gordon
Ford Research
{aarvani1,ksantha4,czeng1,agordo57}@ford.com

Abstract

*We live in a connected world that requires us to identify ourselves every time we want to access our emails, work stations, bank accounts, health care records, etc. Every system we interact with requires us to remember a username/password combination, have access to some private/public key pair, a hardware token, or some third party authentication software. Our digital identity is owned by the services we are trying to access, no longer under our control. Self-Sovereign Identity promises to give back control of his or her identity to the user. It is in this context that we explore the use of biometrics in order to empower users to be their own passwords, their own keys, their own means to authenticate themselves. We propose Self-Sovereign Biometric IDs (*SelfIs*), a novel approach that marries the concepts of decentralization, cancelable biometrics, bloom filters, and machine learning to develop a privacy-first solution capable of allowing users to control how their biometrics are used without risking their raw biometric templates.*

1. Introduction

Online banking, virtual doctor visits, entertainment, and all sorts of digital interactions require we tie ourselves to one of many online unique *digital identities* (e.g., username/email and password combinations). Privacy-aware systems may require the generation of private/public key pairs which may be linked to an account for authentication purposes (e.g., github.com). Trying to manage identities then becomes a challenge that will only grow as more services are digitalized [40]. On one end, we are seeing the centralization of identitity services, where we can log into our favorite website with our social media credentials (e.g., using OpenID [11] or OAuth 2.0 [33]), thereby reducing the number of credentials needed. Some of the downsides of this approach include: 1) losing access to our main account will lock us out of all our linked accounts, 2) we have no visibility into security/insecurity practices of the companies managing our accounts [24], 3) malicious actors could compromise one authentication service and have access to all of our linked accounts. Two-factor authentication is another layer of security with its own pros and cons (e.g., losing a second factor).

Blockchain technology [31] has introduced the concept of decentralized identity to the mainstream. Bitcoin [31], for instance, relies entirely on the idea of decentralized identity, where each user generates a public/private key pair and all interactions on the Bitcoin network are done through the use of Bitcoin addresses, which are derived from the users' public keys. This is very appealing as users do not need to create passwords/accounts to interact with other users, instead, this is seamlessly done by their mobile Bitcoin wallets. The benefits of a decentralized identity management system are manyfold, including an always on notion, fully user controlled, on top of the built-in privacy/security primitives.

Biometric technology aims to reduce the burdens in today's segmentated identity solutions [23, 8], whereby the user becomes his/her own password, because unlike passwords, keys, tokens, or access codes, biometrics can answer the question: *who am I really talking to?*

Our main contributions are: 1) Introduced the concept of Self-Sovereign Biometric IDs (*SelfIs*), which are cancelable biometric templates fully managed by the user. 2) A *novel* machine learning approach capable of extracting features from encoded cancelable bloomed biometric templates.

2. Motivation

Our goal is to provide a way to use biometrics in a secure, privacy-first way, without having to compromise matching accuracy. We want to be able to integrate it into the same fabric as that of existing multi-factor authentication schemes. For that, we will look to extend the concept of Decentralized Identifiers [15]. Although two-factor authentication, biometrics, decentralized identity schemes such as PGP, Bitcoin [31], and SecureKey [20] are very promising, they all address only part of the spectrum of identity. Multi-factor authentication solutions require two things: something you know, and something you have. Passwords, PINs, security questions are all things you know, while private keys and access codes or tokens are things you have. One of the hottest authentication form factors today are biometrics [23, 8]. Rather than being something you know, or something you have, biometrics are *something you are*, and unique to a person [22]. By adding biometric sensors to mobile devices the reach of biometric authentication technology is magnified as mobile technology accounts for over 75% of internet use [46]. One of the biggest issues with biometrics is that once an attacker gets access to the biometric templates, there is nothing stopping an attacker from carrying out a replay attack. Biometrics are especially at risk when centralized [34] as proven by major recent hacks [14, 26].

3. Related Work

3.1. Decentralized Identity

Pretty Good Privacy (PGP) [48] wanted to build a web of trust, where users would have access to keyrings, and prove their identity via PGP key ownership proofs. Bitcoin and many new initiatives borrowed from these concepts to further the idea of decentralized identity. The Bitcoin protocol requires participants in the network to generate private and public key pairs in order to transact within the Bitcoin network. Bitcoin uses Bitcoin addresses, which are derived from the user's public key using a one way function transformation. Consider a simple transaction between Bob and Alice, Bob sends a Bitcoin to Alice's $receive$ address. Alice will then be able to spend that Bitcoin by proving that she owns the private key that is associated with that public key address. Namecoin [32] was one of the first to provide a directory like service, where users could attach identity information such as PGP, OTR keys, email, Bitcoin, and Bitmessage addresses to an identity of their choice. HYPR [18] allows for password-less authentication via biometrics, which are stored locally on their hardware device, however, it requires each user to carry hardware dongles, which is a challenge since any sort of dongle is hard to manage and easy to lose [42]. Like HYPR, SecureKey [20], and many others who use tokens to authenticate users can prove only one thing: *You have access to the keys associated with your identity rather than proving you have your biometrics.* Given the potential of decentralized identity, there are many organizations now trying to define best practices and standards [9, 16].

3.2. Secure Biometrics

Biometric technology has survived the test of time [7, 29, 21]. Traditionally used by law enforcement agencies [29], civil/criminal court, fraud [21], financial industry, etc., it is now mainstream. India and the UAE have major national programs [35, 2]. The academic community has extensively explored ways to protect biometric data. One scheme involves using biometrics to generate keys [39, 28, 17, 1, 43, 10], which are then used to authenticate users, sign documents, etc. Hybrid cryptosystems use a combination of biometrics and crypto primitives to protect data [41, 30]. Cancelable biometrics is a very promising field of research and the academic world has made significant strides since its inception [36, 38]. Cancelable biometrics work by introducing a sort of noise or applying some transformation to the biometric data. This same transformation is then applied at matching time. The key benefit here is that if an attacker steals the transformed biometric data, he or she cannot carry out replay attacks as the template itself has a one-way function applied to it, which in theory is irreversible. More recently, bloom filters have been successfully applied to masking the different biometrics and encoded template matching has been done with promising results [37, 13, 27, 25]. The challenges with these schemes are manyfold: 1) biometric sam-

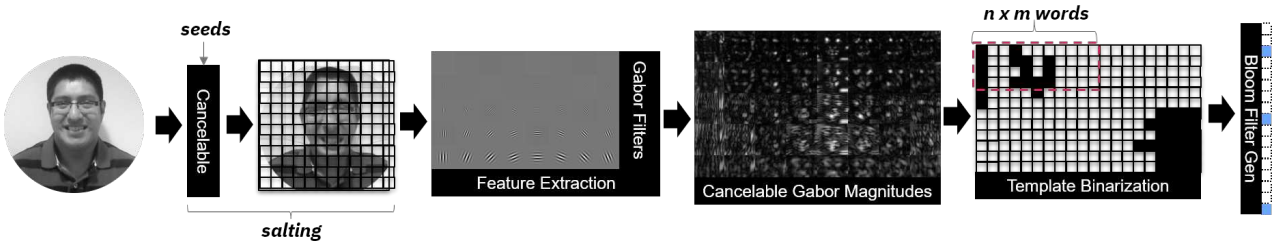

Figure 1. Bloom Filter Generation

pling is nondeterministic, 2) quantization steps lead to quality loss, 3) noise is introduced to the biometrics matching process, all necessary steps to protect biometrics at the cost sacificing matching rates.

4. *SelfIs*: Self-Sovereign Biometric IDs

4.1. Overview: From $selfies$ to $SelfIs$

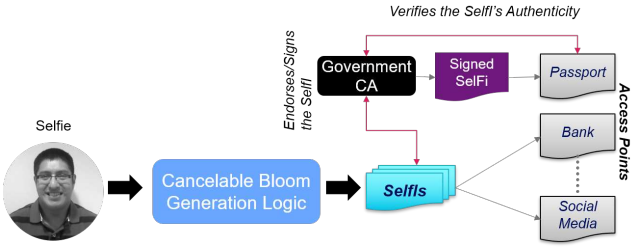

Figure 2. *SelfIs* Uses

SelfIs are simple forms of Biometric IDs that behave similarly to passwords in the sense that passwords can be changed on demand. Moreover, like $secure$ passwords, *SelfIs* are salted before being stored anywhere by using $seeds$ (random numbers used to deterministically transform *SelfIs*). Users can choose to create temporary *SelfIs*, which may have expiration dates, or used by different entities. This is useful for things like physical access control where the use of a *SelfI* can be highly regulated or accessing sensitive data, whereby the data $owner$ and the *SelfI* owner are immediately notified. Any system trying to use a *SelfI* will request two things from the owner: 1) prove he/she owns the $seeds$, and 2) provide a *SelfI* for matching. Figure 2 shows a high-level diagram depicting a use case where a user generates multiple *SelfIs* for interacting with different entities. This use case will require management of some necessary metadata, and we envision an application managing *SelfIs* to behave similarly to a Bitcoin wallet.

4.2. Cancelable Bloom Generation

Our work extents [36, 13] in that we 1) salt biometric templates in order to allow for secure storage of the templates and 2) apply one-way functions to biometric data in order to mask the original templates. First, we take a $selfie$, a set of random numbers are generated (the $seeds$), which dictate how the image is transformed and are stored locally under user control. The second step is the blooming of the cancelable template. Once the blooming is done, we then store the bloom data as well as hashes of the seeds in a Blockchain fabric such as Hyperledger Fabric [6]. Figure 1 shows the different stages an image goes in order to extract a set of bloom filters. First, the image is decomposed into feature vectors of n-blocks, which are then rearranged and $n-bit$ by $m-word$ blocks are extracted from the vertical group arrangement. These blocks are then encoded in 2^{n-bits} bloom filters. Which are then stored for matching purposes. Bloom filters can be further encoded and protected through the introduction of noise to the process, which leads to higher degree of security. One of the biggest benefits here is that even if the bloomed data is compromised, the transform is irreversible. Once a $selfie$ has made it through the Cancelable Bloom Generation process, a *SelfI* is generated. Listing 1 shows a sample object generated and sent to the backend for registration, where $data$ consists of either a single cancelable bloomed template for authentication purposes, or an array of bloomed images for registration purposes. We integrate *SelfIs* with Decentralized Identifiers ($DIDs$), which are a new type of identifier for verifiable digital identity [15]. Each DID Document contains at least three things: cryptographic material, authentication suites, and service endpoints. Cryptographic material combined with authentication suites provide a set of mechanisms to authenticate as the DID subject (e.g. public keys, pseudonymous biometric templates, etc.).

Service endpoints enable trusted interactions with the DID subject [15]. Once the registration step is complete, the DID belonging to the user is updated with the necessary data for matching. Listing 2 shows an example of a DID corresponding to a *SelfI*.

Listing 1. Registration JSON

```
{
   "user": "danny",
   "type": "register",
   "uuid": "2b84def1-02e0-48a3-9af5-29d134da8960",
   "data": ["A04B8081A08080088800...",
               "8001800008000800080000...",
               "9309A000A040C1008001.."],
   "nonce" : "1542140591.9916613",
   "address" : "1A4aNCeC1Zt2BPHX8JDQtjPUgwvuDY4uxp"
       ,
   "signature" : "HzPe9HSVT74SqVZgf..."
}
```

Listing 2. Sample DID Object

```
{
   "@context": "https://w3id.org/did/v1",
   "id": "did:selfi:2b84def1-02e0-48a3-9af5-29d134da89
       60",
   "authentication" : [{
      "id": "did:selfi:2b84def1-02e0-48a3-9af5-29d134
          da8960#facial",
      "type": "cancelable_face",
      "controller": "did:selfi:1A4aNCeC1Zt2BPHX8
          JDQtjPUgwvuDY4uxp",
      "seed_kx_hash": "4deb118d25fc29d51ba2f07...",
      "seed_ky_hash": "abc6eb9a6075f3af43418b0..."
      "data": ["A04B8081A08080088800...",
                "8001800008000800080000...",
                "9309A000A040C1008001.."],
      "ca_info": {
         "signature": "HzPe9HSVT74SqVZgf...",
         "entity": "did:v1:abbc2ab9-467e-4a80-87d5-df4
             d904ca2af"
      }
   }],
   "service": [{
      "type": "AuthenticationService",
      "serviceEndpoint": "https://host.auth.service.
          com/api/v1/match"
   }]
}
```

4.3. Enter the Blockchain

By using Blockchain we have decentralization for distribution and matching of templates, as well as crypto primitives built into the fabric to transfer data in a secure way. If a user wants to prove to someone that he/she owns the seeds to perform the one-way transform for the biometric template, the user can simply sign the hash of the seeds that are associated with the bloomed data. Signing the hash of the seeds tells someone that the user owns the seeds associated with the bloom data. Similarly, if a user is challenged and asked for the seeds in order to do a match, the user can choose wether to answer that challenge and provide his/her template's seeds or not. Distributing seeds can be done via PGP/Public Key cryptography, whereby, the user knows the $address$ or $PubKey$ of the person/system issuing the challenge.

4.4. The Access Point

Once the seeds are distributed, the *access point* will capture the user's biometric template, apply the seeds for the cancelable transformation, and proceed with the blooming. We assume the *access point* is $trusted$ (e.g., implemented with ARM TrustZone [3]). If at any point in time an *access point* is compromised, then the user's raw biometrics may be compromised as well, which is a problem with any biometric solution.

4.5. Machine Learning

At the core of our solution is a novel Deep Learning-based approach that extracts and learns features/patterns from cancelable bloom data (*SelfIs*). This step will be described in more detail in Section 5.

4.6. Use Case: Traveling with *SelfIs*

4.6.1 The Requirement

Consider the scenario where a user is to visit a foreign country and needs to do some banking, social media, get through customs, etc. The user brings his/her smart phone which can be used to satisfy the two-factor requirements of many systems. Let's further assume that digital passports are accepted. In this context, if the user has access to his/her phone we are good. Now, consider a scenario where the user does not have immediate access to the phone (e.g., it is lost, broken, no WiFi password, out of battery, etc.). In this context what would be the next best thing? Biometrics? Let's assume the country abroad uses Biometrics to do things like banking, get through customs, etc. Now, the next question is, do we trust a foreign country to properly manage our biometrics [14, 26]? Hence, the requirements: 1) Allow for multi-factor (possibly password, PIN, some token multiple types of biometrics, etc.),

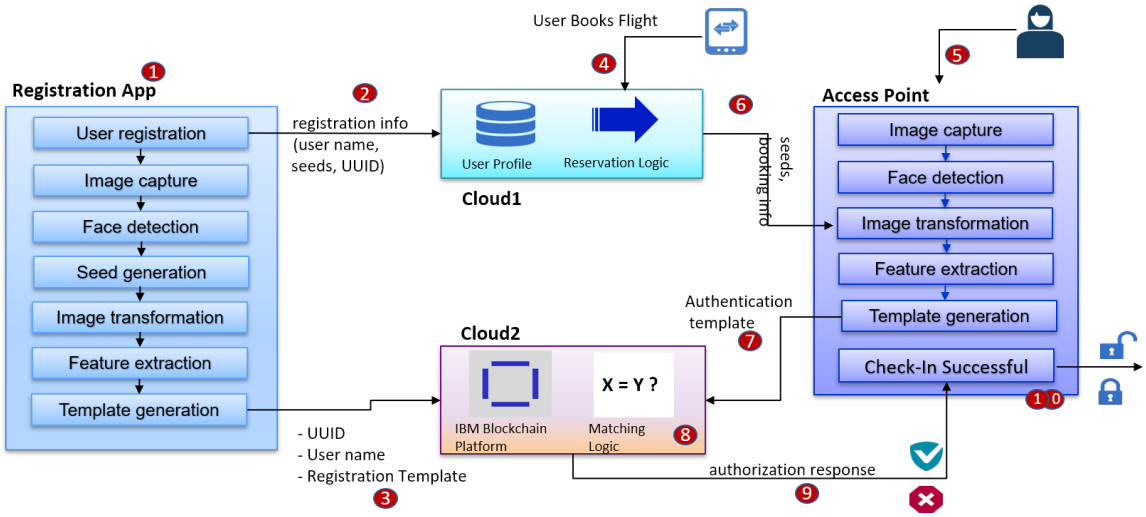

Figure 3. *SelfIs* Architecture

2) If we lose any of our factors, we should still be able to authenticate.

4.6.2 The Multi-Factor Solution

Before traveling we will register ourselves as well as our authentication factors by extending $DIDs$ [15] to incorporate authentication factors such as PINs, *SelfIs*, etc. Figure 3 shows a high-level diagram of our current prototype for registration of *SelfIs*. For things like passwords, PINs, etc., things we know, we can choose to create temporary tokens that can be deterministically derived from our PINs/passwords. We can use these to authenticate ourselves, we can register them as authentication factors with $PubKey$-derived addresses in our Blockchain (e.g., linked to a DID). For tokens, or access codes, we can assume we have our phone, thus, we can use those factors to authenticate ourselves. The key advantage of using *SelfIs* is that we can lose our other authentication factors and still be able to use our Biometrics. In this context, we could then use our approach to support *n-of-m* factor authentication depending on what we want to access. For instance, if we want access our Bank account and get some cash out, the Bank may ask for a PIN and a *SelfI*. Both of which can can use the same infrastructure. The bank would pull the $PubKey$ associated with the PIN part of our DID, use the $PubKey$ to verify a challenge token signed with the $Private\ Key$ derived from our PIN. The *SelfIs* would then be used as a second authentication factor.

4.6.3 Traveling with Agency-backed *SelfIs*

Users can use a mobile Application to register their *SelfIs* with a trusted Backend of their choosing (1). For instance, in Figure 3, a user is going to travel abroad, thus, he/she needs to register their *SelfIs* with the Airline ($Cloud1$). The Airline may choose to trust the the Matching Service running in $Cloud2$, or it could ask the user to use another $trusted$ Matching Service provider. The user will then register his/her *SelfI* with the Airline, and provide among other metadata, the necessary seeds to derive the Cancelable Bloom Data (2). The seeds are distributed on demand, however, in this use case, we assume the seeds are stored in the Airline's backend ($Cloud1$). The *SelfIs* are added to the Blockchain fabric and Machine Learning backend ($Cloud2$) for training purposes so that the user can be authenticated at a later point in time (3).

When the user books a flight, the Airline will create the necessary booking information (4). Next, the user goes to Check-In and the automated kiosk (an *access point*) (5). The Check-In kiosk then pulls the necessary metadata from $Cloud1$ to generate the matching *SelfI* (6), and makes the call to the Matching backend (7) in $Cloud2$. The matching logic is triggered and a response is securely (e.g., SSL) sent back to the *access point* (8, 9). The Check-In process can now continue (10).

In the event that a $trusted$ *SelfI* is required, a third party such as a Government Agency can $endorse$ *SelfI*. This process is similar to the one above meaning that, when a user generates a *SelfI*,

the user will follow the same steps as the use case above, generate the *SelfI*, provide the seeds to the Government agency, then let the Government agency generate a *SelfI* for the user and match it. If there is a match, the Government agency can then $endorse$ the user's *SelfI*. This is analogous to the process of digital certificates, whereby upon proving ownership of a *SelfI* and its contents, the Certificate Authority $endorses$ it, and we can now use standard methods to verify authenticity of a *SelfI*.

5. Model and Experimental Results

5.1. Architecture Software Stack

We developed an Android Application that is used to capture the user's selfies and register/authenticate with our backend. IBM Cloud hosts our booking and machine-learning logic. IBM's Blockchain Platform [19] is used to manage our user's *SelfIs* via RESTful endpoints [4]. We built a native library for *SelfI* generation that supports ARM and x86 platforms. We currently support Android, Raspberry Pi 3, and x86 $trusted$ *access points*.

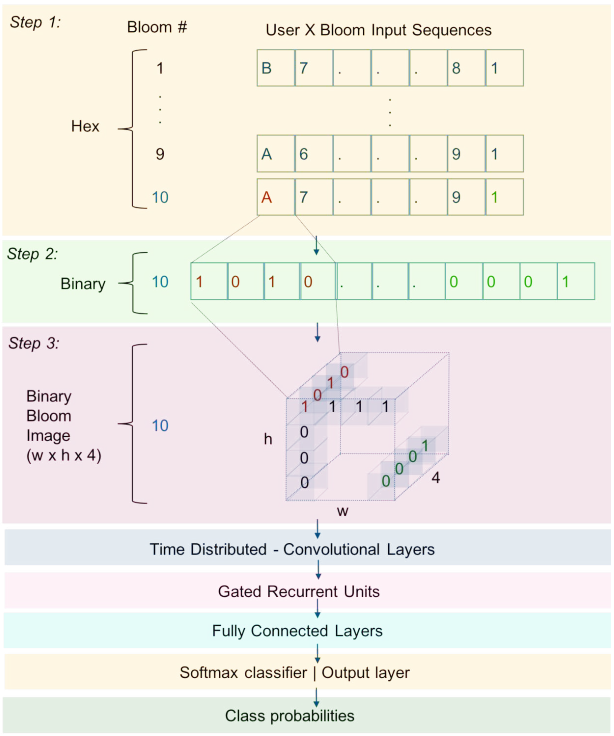

Figure 4. Overview of our model architecture

5.2. Template Matching Model

Convolutional neural networks (CNNs) have been studied and used extensively to do image classification. The task is to assign a label or class to an input image and provide a confidence score or probability that the input image belongs to a particular class. In order to train these types of networks, a large number of training samples per class are required, and the data typically consists of RGB images with 3 channels. In the area of time series classification, deep convolutional neural networks have been studied to demonstrated that feature learning is possible for time series classification [47]. One of the reasons is that the use of very long sequences in recurrent neural networks such as Long short-term memory (LSTM) is difficult because it is a very tedious task to train these models due to the length of the sequences. Furthermore, using sequential models like Dynamic Time Warping (DTW) [44] or Hamming distance and variants of it [5] to measure the similarity between two sequences either takes too long to execute or might not be able to provide the level of accuracy required to do template matching. Inspired by the efforts to do and improve image classification as well as the on-going research to apply deep learning to DNA classification and gene regulation [12, 45], we devised our own model to do bloom classification. Our model takes into account the spatial and temporal nature of the hexadecimal bloom sequence by encoding the bloom filters as images and using various layers to learn features from the bloom filters. The model consists of time distributed convolutional layers followed by Gated Recurrent Unit (GRU) and fully connected layers. Figure 4 shows a high level overview of the architecture. The intuition behind the proposed model is to convert hexadecimal bloom filters into images with 4 channels. Each channel holds a single bit, which dictates if a feature is present or not for a given bloom template. The convolutional layers are used to learn high level features/patterns. The data is then flatten and fed to a set of GRU and fully connected layers. This allows us to extract and learn additional features and train the model at a faster rate. The activation used throughout the layers is ReLU. Max pooling and batch normalization are used to extract additional features, reduce overfitting, and use of a higher learning rate. The softmax activation function is used in the final layer to get class probabilities.

5.3. Data Sources and Pre-processing Steps

In order validate the proposed model, we created our own dataset of face images taken from a Sam-

sung Galaxy S9 Plus device. The dataset consist of 400 images of size 2448 x 3264 pixels taken under different lighting, background, and face alignment conditions. In total 20 images were taken per participant, stored in JPEG format, and labeled with an identification number. This dataset is representative of the registration phase mentioned in the Traveling Abroad use case since each person is required to use his or her smartphone to take *SelfIs*, and the participants take the images under different lighting and background conditions. In addition, we used some of the images from Caltech's Frontal face dataset as well. For the experiments and results reported in this paper, we use the images from the dataset we have created for training, testing, and validation. Images from the Caltech dataset as well as some images we took with a different device are used for testing purposes only since the number of images available per person in this dataset varies a lot and we need at least 10 images per person to train our model.

For the images in the dataset we have created, we randomly split each set of 20 images into training, testing, and validation sets. Of the 20 images taken for each participant, 10 images are used to train our model, 5 images are used to validate our model, and the remaining 5 images are used for testing. For the Caltech's dataset images and other images we took, we randomly selected 2 images per participant and use them only for testing scenario 4, described in Section 5.4. Upon splitting the data, we followed the steps mentioned in the use case to detect the face, transform (distort) the face, extract features from the face region of the image, and generate the bloom filters. Each bloom filter consists of 61,440 hexadecimal values that encode the feature vectors. We then iterate through each hexadecimal value to convert it to its binary representation (4 bits per hexadecimal value) for a total of 245,760 bits. Finally, each binary bloom filter of size *1 x 245,760* is converted into an image representation of size *w x h x c*, where *w* is the width of the image, *h* is the height of the image, and *c* is the number of channels/depth. The resulting binary bloom image is of size *240 x 256 x 4*. Each element of the binary bloom image represents the hexadecimal value in binary format. Figure 4 shows an example of the transformations that takes place to convert a hexadecimal bloom filter to its corresponding binary bloom image representation. Step 1 shows the 10 blooms of a user, each in hexadecimal format. Step 2 shows the binary representation of the 10th hexadecimal bloom filter. Step 3 shows the representation of the bloom filter as a binary bloom image format. The transformation from hexadecimal bloom filter to binary bloom image is done for all bloom filters for all participants.

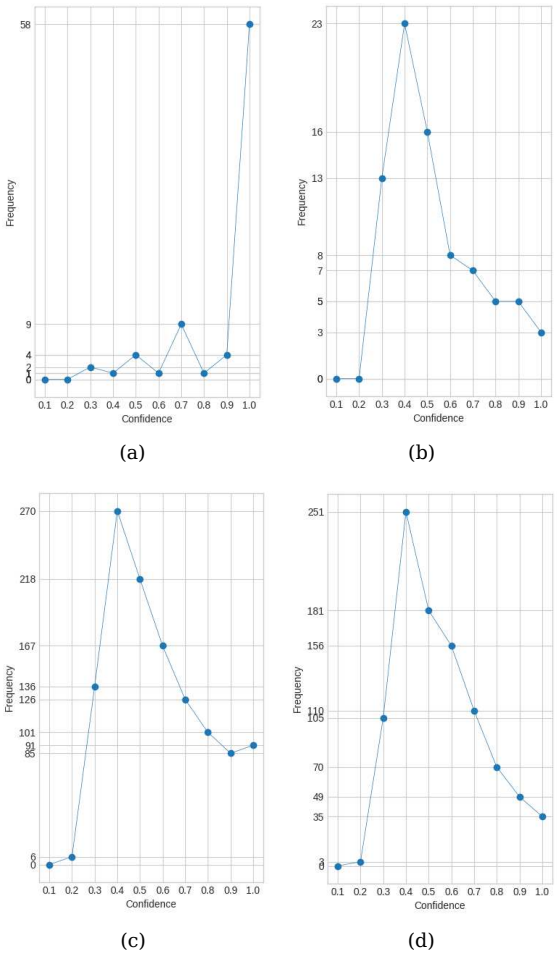

Figure 5. Distribution of confidence levels for the 4 scenarios described in section 5.4: (a) Scenario 1: test using same seeds used during registration; (b) Scenario 2: test using randomly generated seeds never used before; (c) Scenario 3: test using seeds of all registered users; (d) Scenario 4: test using seeds of all registered users on users that have not registered before.

5.4. Experiments and Results

We conducted a set of experiments to test 4 different scenarios that are likely to occur when performing face authentication using the *SelfIs* architecture shown in Figure 3. For the following scenarios, let U_x be a person who has registered using the *SelfIs* system, S_x be the seeds used to transform U_x images, and U_y be a person who has not registered.

This can be extended to all users who have registered, but for simplicity, the following scenarios describe the interaction between our system and users U_x and U_y. *Scenario 1*: Given the seeds generated for U_x during the registration phase, use U_x seeds, S_x, to bloom U_x images for authentication. This describes the case in which U_x wants to authenticate. *Scenario 2*: Given some randomly generated seeds that are not equal to S_x, use these random seeds to authenticate U_x. This describes the case in which U_x wants to authenticate but the system performs a different transformation since it is using some random seeds that are not S_x. *Scenario 3*: Given the seeds generated from all other registered users, excluding S_x, use these seeds to authenticate U_x. This describes the case in which the system is waiting for U_x to authenticate, but a different user who has registered with the system tries to authenticate. *Scenario 4*: Given U_y, who has not registered, use the seeds from all registered users to authenticate U_y. This describes the case in which a user who has not previously registered tries to authenticate.

We performed the pre-processing and dataset splits described in section 5.3 to build our training, validation, and testing sets for all 4 scenarios. Out of the 20 sets of data from the dataset we created (20 participants, 20 images each), 16 sets are used for training, validation, and testing scenarios 1-3. The remaining 4 sets from this dataset plus images from the Caltech dataset and the images we took from a difference device are used to test scenario 4 since this scenario requires images of participants that the system has not seen before. We set the cutoff confidence threshold to 80% to indicate a match if the confidence level is equal or greater than 80%; a mis-match if the confidence level is less than 80%. The results are summarized in Table 1. For scenario 1, 62 out of the 80 predictions (77.5% of the predictions) have a confidence score equal or greater than 80%. The remaining 18 predictions (22.5% of the predictions) have a score less than 80%. Scenarios 2-4 have a high percentage of confidence levels below the cutoff threshold as one would expect, 90.0%, 85.33%, and 91.25% respectively. For scenarios 3-4, the number of predictions are much larger due to the number of possible seed combinations. Figure 5 shows the distribution of confidence levels across all 4 scenarios. The distributions show that the vast majority of confidence levels for scenarios 2-4 are within confidence levels 0.3 and 0.5. For scenario 1, 58 predictions have a confidence level between 0.9 and 1.0, 4 between 0.80 and 0.89, and so forth. These results indicate that our model learns from the binary bloom images and is able to authenticate (match) a large number of users if the same set of seeds generated during the registration phase for that particular user are used during the authentication phase. Otherwise, the system should not match blooms if the features from the users are different or if different transformations are applied to the data.

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
# predictions	80	80	1200	960
# Above cutoff	62	8	176	84
# Below cutoff	18	72	1024	876
% Above cutoff	77.5%	10.0%	14.67%	8.75%
% Below cutoff	22.5%	90.0%	85.33%	91.25%

Table 1. Results summary for all 4 scenarios described in section 5.4.

We also performed some optimization and tried different models (different type and number of layers) during our experiments. These variations showed higher matching rates (% above cutoff) for Scenario 1 (e.g., 82.5% and 85% for two different tests), but also higher false positive rates for Scenario 2 (17.5% and 16.25%), Scenario 3 (16.0% and 13.5%), and Scenario 4 (6.46% and 13.23%). The above cutoff percentage rates for Scenarios 2-4 should be close or equal to 0%, or conversely, the below cutoff percentage rates should be close or equal to 100% to indicate that there is no match when in fact there should not be a match. Furthermore, we deployed the models and trained them with live data, but we also observed lower accuracy results. This leads us to believe that we need to fine-tune our models further to account for different characteristics such as different face orientations, lighting conditions, etc.

6. Conclusion

This paper proposed the concept of Self-Sovereign Biometric IDs (*SelfIs*). The idea is to allow users to fully control how their biometrics are used for the purpose of digital identity. We show preliminary results describing the use of machine learning combined with cancelable biometrics, which show promising acceptance rates. Future work includes hyper parameter optimizations, increase our data set to include images from multiple platforms, as well as integrate mixed-mode blooming (other biometrics).

References

[1] The Practical Subtleties of Biometric Key Generation.

[2] Uae government adopting facial and iris recognition to identify customers.

[3] ARM. Arm trustzone technology.

[4] Luis Bathen. Ibm blockchain platform rest connector.

[5] Abraham Bookstein, Vladimir A Kulyukin, and Timo Raita. Generalized hamming distance. *Information Retrieval*, 5(4):353–375, 2002.

[6] Hyperledger Community. Hyperledger fabric. 2017.

[7] Swiss Confederation. The fingerprint: 100 years in the service of the swiss confederation. 2013.

[8] K. Delac and M. Grgic. A survey of biometric recognition methods. In *Proceedings. Elmar-2004. 46th International Symposium on Electronics in Marine*, pages 184–193, June 2004.

[9] (DIF). Decentralized identity foundation. 2017.

[10] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *SpringerLink*, pages 523–540. Springer, Berlin, Heidelberg, May 2004.

[11] OpenID Foundation. Openid.

[12] Xin Gao, Jie Zhang, Zhi Wei, and Hakon Hakonarson. Deeppolya: A convolutional neural network approach for polyadenylation site prediction. *IEEE Access*, 6:24340–24349, 2018.

[13] Marta Gomez-Barrero, Christian Rathgeb, Javier Galbally, Christoph Busch, and Julian Fierrez. Unlinkable and irreversible biometric template protection based on bloom filters. *Inf. Sci.*, 370(C):18–32, Nov. 2016.

[14] Andy Greenberg. Opm now admits 5.6m feds' fingerprints were stolen by hackers. 2015.

[15] Credentials Community Group. Decentralized identifiers (dids) v0.9. 2017.

[16] Working Group. Hyperledger identity working group. 2017.

[17] Hao Feng and Chan Choong Wah. Private key generation from on-line handwritten signatures. *Information Management & Computer Security*, 10(4):159–164, Oct. 2002.

[18] HYPR. Decentralized biometric authentication. 2017.

[19] IBM. Ibm blockchain platform.

[20] SecureKey Technologies Inc. Digital identity network. 2017.

[21] A. Jain and S. Pankanti. Beyond fingerprinting: Is biometrics the best bet for fighting identity theft? 2008.

[22] Anil K. Jain, Sharath Pankanti, Salil Prabhakar, Lin Hong, and Arun Ross. Biometrics: A grand challenge. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 935–942, Washington, DC, USA, 2004. IEEE Computer Society.

[23] Anil K. Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Trans. on Circuits and Systems for Video Technology*, 14:4–20, 2004.

[24] Brian Krebs. Facebook stored hundreds of millions of user passwords in plain text for years.

[25] Yen-Lung Lai, Zhe Jin, Andrew Beng Jin Teoh, Bok-Min Goi, Wun-She Yap, Tong-Yuen Chai, and Christian Rathgeb. Cancelable iris template generation based on Indexing-First-One hashing. *Pattern Recognition*, 64:105–117, Apr. 2017.

[26] Ashish Malhotra. The world's largest biometric id system keeps getting hacked. 2018.

[27] Edlira Martiri, Marta Gomez-Barrero, Bian Yang, and Christoph Busch. Biometric template protection based on Bloom filters and honey templates. *IET Biometrics*, 6(1):19–26, Jan. 2017.

[28] F. Monrose, M. K. Reiter, Qi Li, and S. Wetzel. Cryptographic key generation from voice. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, pages 202–213, 2001.

[29] R. Moses, P. Higgins, M McCabe, S. Probhakar, and S. Swann. Automated fingerprint identification system (afis). 2010.

[30] Abhishek Nagar, Karthik Nandakumar, and Anil K. Jain. A hybrid biometric cryptosystem for securing fingerprint minutiae templates. *Pattern Recognition Letters*, 31(8):733–741, June 2010.

[31] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[32] Namecoin. Namecoin. 2011.

[33] OAuth. Oauth 2.0.

[34] The Times of India. Aadhaar linking risky, makes it easy prey for hackers. 2018.

[35] Unique Identification Authority of India. Aadhaar. 2015.

[36] V. M. Patel, N. K. Ratha, and R. Chellappa. Cancelable biometrics: A review. *IEEE Signal Processing Magazine*, 32(5):54–65, Sept 2015.

[37] C. Rathgeb and C. Busch. Cancelable multi-biometrics: Mixing iris-codes based on adaptive bloom filters. *Computers & Security*, 42:1–12, May 2014.

[38] Andrew B.J. Teoh, Yip Wai Kuan, and Sangyoun Lee. Cancelable biometrics and annotations on BioHash. *Pattern Recognition*, 41(6):2034–2044, June 2008.

[39] George J. Tomko, Colin Soutar, and Gregory J. Schmidt. Fingerprint controlled public key cryptographic system, July 1996. U.S. Classification

380/30, 380/54, 713/186, 380/285; International Classification H04L9/30, H04L9/08, G06K9/00, G07C9/00, G07F7/10; Cooperative Classification H04L9/0866, G07C9/00087, G06K9/00006, G06Q20/341, G07F7/1008, G06Q20/40145, H04L2209/805, H04L9/0662; European Classification G06Q20/40145, G06Q20/341, G06K9/00A, H04L9/30, G07C9/00B6D4, H04L9/08, G07F7/10D.

[40] J. Torres, M. Nogueira, and G. Pujolle. A survey on identity management for the future network. *IEEE Communications Surveys Tutorials*, 15(2):787–802, Second 2013.

[41] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain. Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, June 2004.

[42] Veridium. Replacing rsa tokens with biometrics. 2017.

[43] C. Vielhauer, R. Steinmetz, and A. Mayerhofer. Biometric hash based on statistical features of online signatures. In *Object recognition supported by user interaction for service robots*, volume 1, pages 123–126 vol.1, 2002.

[44] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040. ACM, 2006.

[45] Haoyang Zeng, Matthew D Edwards, Ge Liu, and David K Gifford. Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127, 2016.

[46] Zenith. Mobile advertising forecast. 2017.

[47] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[48] Philip Zimmermann. *The Official PGP User's Guide*. The MIT Press, Cambridge, MA, 1995.