Towards a more secure Aadhaar

Ajinkya Rajput[1] and K. Gopinath[1]

Indian Institute of Science, Bangalore India
ajinkya,gopi@iisc.ac.in

Abstract. Aadhaar is the national identities project of the Government of India. The main benefit of Aadhaar is expected to be better decision making using modern analytics, as citizens use such an identity to avail of services from various government as well as private service providers; this necessarily involves building a huge store with necessary information on citizens such as mapping of ids to biometrics. Such a store raises many security and privacy concerns and therefore should be designed and analyzed very carefully. The threat model for such systems should address both internal and external attackers. Previous writings and research work[12] in this area have discussed problems such as illegal profiling and tracking of individuals, authentication without consent, collusion of multiple service providers leading to correlation of user data, and use of fake biometrics. While some analyses have focussed on cryptography to provide a solution, a comprehensive and workable solution for, say, illegal profiling, is still lacking, and there are also many problems from a systems perspective that need to be addressed such as access control models to constrain the access to sensitive data as well as integrity of its metadata. In this paper, we discuss solutions to such problems, esp illegal profiling.

Keywords: National Identities, Privacy Preserving Identities, Aadhaar

1 Introduction

The national identities in India are provided by the central government under THE AADHAAR (TARGETED DELIVERY OF FINANCIAL AND OTHER SUBSIDIES, BENEFITS AND SERVICES) ACT, 2016 [5]. Under this act, the government has established the Unique Identification Authority of India (UIDIAI), which takes care of the operation of the Aadhaar system. Aadhaar is now possibly the world's largest biometric database with 115+ crore identities stored along with biometric information[14]. During enrollment process under Aadhaar, citizens of India submit their demographic information including name, address, gender and year of birth along with their biometric information including fingerprints, iris scans, and a photograph. Each citizen is then issued a unique 12 digit Aadhaar number which ties all this information together.

1.1 Pros of Aadhaar

The central government and many states in India are using the Aadhaar number for direct transfer of various subsidies to citizens and this has reportedly reduced

fraud[11] on a large scale. The Aadhaar number can be used to maintain and link census, immunization records, etc. and thus may empower a government to use modern data mining and analytics techniques to make informed policy decisions. Likewise, Aadhaar may also help criminal investigations across the country. In cases of natural disasters or mishaps like train accidents, the Aadhaar system may help in identifying victims for immediate care.

Private sector service providers like banks, telecommunication companies etc. also need to verify the identification information provided by their clients; the Aadhaar system provides this infrastructure. Aadhaar also provides an eKYC API, by which service providers can get identification data directly from Aadhaar database for any Aadhaar number. This is subject to authentication from the client using OTP/Biometrics. This greatly reduces time and effort required to avail services.

1.2 Challenges with Aadhaar

However, there are huge risks involved with maintaining a large store of biometric information of all the citizens. A huge centralized database in itself is a problem as a compromise would lead to leakage of data of a large number of citizens. There have been many reported cases, for instance, when hackers allegedly compromised the national identities database of Turkey and stole about fifty million fingerprints [3]. This data can be used for many malicious purposes such as identity theft, financial frauds, tracking of individuals etc. The Aadhaar project has been and is being seriously debated due to privacy and security concerns. One major concern is that of mass surveillance, as government agencies can use Aadhaar system to track or profile individuals without proper warrants. Note that the problem is not trivial as an Internet-wide PKI system. Internet-wide PKI is a much simpler system (it "only" maps users to public keys and does not use biometrics) but even such a system has not been easy to provision or use due to revocations, failures, and disconnections.

Aadhaar numbers have been linked with bank accounts (for example, in UPI applications). Income Tax department of India uses Permanent Account Number (PAN) to identify each tax paying entity. The government of India has made it mandatory to link Aadhaar number and PAN. This makes the security of the whole system more critical since now the finance sector can be targeted by malicious parties. Identity thefts using Aadhaar can now be used in tax frauds.

Also, there have been at least three instances where the ill-designed websites of state government schemes have leaked millions of aadhaar identities [1] with one of them leaking 130M of them. These leaked data can be used in a range of malicious activities. In current practice, Aadhaar ids are collected by all the service providers as a part of "Know Your Customer" requirement. Multiple service providers can collude together or with employees of UIDAI to profile customers and then can abuse/sell this information for profit. This is clearly unethical and the UIDAI should have safeguards in place to prevent such practices.

The Aadhaar act mentions that the system should be secure but does not mention any technical specifications for security and privacy for the system.

UIDAI on its website does mention the security measures it has taken but its effectiveness is not clear. Published information indicates that end-to-end encryption is used for security and therefore sufficient. But this is clearly not enough as neither privacy issues can be handled by this nor can one be sure that there are no other ways to attack the system (for example, compromise through HTTPS interception). Another example of the inadequacy of the measures taken by UIDAI is the use of Hardware Security Modules to handle keys. But HSMs under same administrative control as the database is not effective. Although the government of India has made provisions for strict penalties in the Act for fraudulent activities of any kind but these are just legal provisions which have effect only if the breach is known and the violator apprehended.

The government plans on linking health sector and voter identification cards with Aadhaar numbers in near future [16]. Linking of health records to Aadhaar risks public safety. A huge amount of infrastructure and operations are going to be dependent on Aadhaar when usage of Aadhaar is expanded in these areas. The issues mentioned above highlight importance of secure system design and careful analysis of Aadhaar. The system design should be, as far as possible "provably" secure with privacy guarantees as necessary for a system as sensitive as Aadhaar.

1.3 Our Contributions

With all the challenges stated in the previous paragraph, the Aadhaar system needs to be analyzed very carefully and thoroughly from a comprehensive systems perspective. Although there have been many careful analyses of Aadhaar system (for example, by Shweta Agarwal et al. [12] from a cryptographic perspective), such analyses still do not have well worked out models for protecting privacy when aadhaar numbers are used across multiple consumer services; neither do they take into account sufficiently the systems perspective. For example, many of the observations in Shweta Agarwal's paper require an access control model and this is not discussed though it is a critical part of the security of the system.

In this paper, we discuss the security aspects of the Aadhaar system and highlight some concerns in the current system such as replay attacks using biometrics, and profiling and tracking of individuals. We put forward our analysis of these problems and propose solutions.

The paper is organized as follows. First, we give a brief outline of the Aadhaar system in section 2. Next, we give an analysis of the current problems in the Aadhaar system and their solutions in Section 3. In section 4 and 5, we present our access control based model of Aadhaar. We present our conclusions in section 6.

2 The Aadhaar Architecture

Figure 1 denotes current architecture of Aadhaar system. Following five entities are part currently part of Aadhaar system.

– **CIDR:** Central Identities Data Repositories (CIDR) is the central database in which all the electronic records are stored. It is managed by UIDAI and responds to verification request with a Yes/No response.
– **AUA:** Authentication User Agencies (AUAs) are third party service providers who require their clients to be authenticated by the Aadhaar system using the Aadhaar numbers of clients. They in turn submit the verification request.
– **ASA:** Authentication Service Agencies (ASAs) are connected with CIDR through leased lines and forward authentication requests to CIDR on behalf of one or more AUAs.
– **Aadhaar User:** Aadhaar users are the citizens of the country who are issued Aadhaar numbers. Their biometric information is stored in the CIDR.
– **Authentication Devices:** Authentication devices are the devices which are used to read biometrics from the users for authentication.
– **Enrollment Agencies:** Aadhaar users need to go to enrollment agencies to register their biometric information in the CIDR. Enrollment agencies are hired by UIDAI to perform these duties. (These are not shown in the diagram.)
– **eKYC API:** Third-party service providers can get an electronic copy of Aadhaar card of the user by invoking this API. This API returns the user data only if the request is authenticated by the user through biometrics or by OTP on registered mobile number. This API is gaining popularity as the service provider has to take only a minimum amount of data from the user such as just the Aadhaar number and biometric authentication.

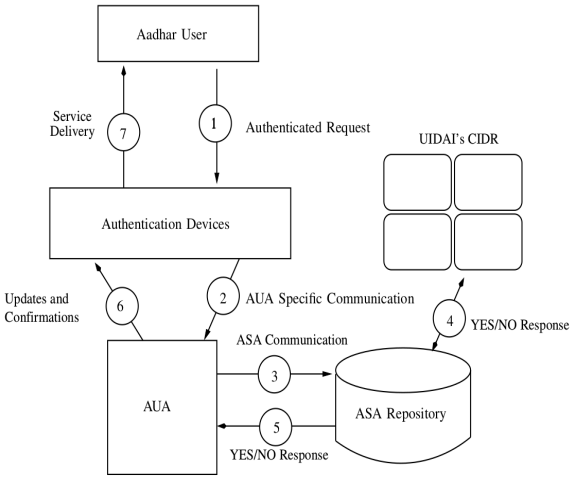

Fig. 1. Architecture of current Aadhaar System

Given the large numbers of actors involved in running the system, a fine grain access control policy is imperative and this is discussed in Section 4.

3 Some Serious Problems in Current Aadhaar System and Their Resolution

Some of the concerns raised by researchers (eg. Shweta Agarwal et al. in [12]) in a system like aadhaar are as follows:

- Confusion with respect to basic design goals such as Identification vs Authentication.
- Identification of individuals without consent using the global Aadhaar number.
- Identification and authentication without consent using demographic and biometric data.
- Surveillance, tracking or profiling of people beyond legal sanctions using the centralized database, either through external hacks or through insider leaks and collusion
- Lack of a formal model of access control for various actors in the system based on roles and attributes, and the integrity of the attribute metadata.
- lack of analysis from a software engineering perspective on a good design architecture.

Next, we analyze each problem and propose our solutions.

3.1 Identification vs Authentication

UIDAI seems confused between identification and authentication; note that identification is just verification of a mapping between different attributes of users, while authentication requires some kind of secret or token. For example, we need identity verification for opening a bank account while we need authentication to make a transaction.

In the Aadhaar Act 2016[5], "authentication means the process by which the Aadhaar number along with demographic information or biometric information of an individual is submitted to the Central Identities Data Repository for its verification and such Repository verifies the correctness, or the lack thereof, on the basis of information available with it". It considers identity verification as authentication. This confusion leads to problems in the design downstream.

3.2 Identification without consent

Another major issue with Aadhaar system is that the biometric data used for authentication is considered to be private, which may not be appropriate. Fingerprints of individuals can be easily lifted from the objects once touched or can be forged easily and printed on prosthetic fingers[17]. Iris scans can be extracted from high resolution photographs even from a distance of six feet [9]. This leads to authentication without consent and larger problems like identity thefts. Hence biometric data without liveness detection techniques should not be used for authentication. Another attack vector on the system is by replaying

the authentication message by service providers. Such cases are reported and UIDAI has already taken action against involved service providers [13].

Liveness detection checks if the current biometric modality being read is from a live human being. There are various hardware based and software based techniques for liveness detection. The hardware based techniques require extra hardware with the biometric reader. They use different parameters like electrical conductivity, perspiration etc. While software based solutions use signal processing and machine learning algorithms, these algorithms use features like ridge frequency, ridge height and other biometric parameters for liveness detection. These parameters usually change when fingerprint are forged on materials used to spoof fingerprints. This change of fundamental parameters can help software solutions to determine liveness of a fingerprint. Software based solutions are cheaper and non-invasive. Hardware based liveness detection prevent replay attacks.

UIDAI does not mandate liveness detection; In the specifications of the UIDAI for fingerprint scanners[15], liveness detection is "recommended but is not compulsory". This still leaves the issues of non-compliance by the AUAs due to the non-use of a liveness detection enabled fingerprint scanner. Ideally, UIDAI should implement software liveness detection at CIDR along with making liveness detection mandatory for fingerprint scanners used by AUAs.

3.3 Preventing Correlation and Illegal Tracking

The problem and its implications Aadhaar number is an unique id provided to each individual. This number is consistent across all the domains and is needed to be given to service providers to avail services. Now, these service providers can collude and track and or profile an individual. They can use this information maliciously or sell this information. If an employee of CIDR colludes with the service providers the severity of problem increases multifold as the data available to the malicious party is much more. There are legal provisions in the Aadhaar Act but by definition, these provisions kick in only after the crime is already committed whereas a system design should be robust and secure enough to prevent such practices as far as possible.

The solution UIDAI has identified this problem and has proposed a solution. The UIDAI solution is a recommendation to all the service providers to use local ids and maintain a mapping from global Aadhaar ids to local ids. This solution does not solve the problem as the service provides still have access to Aadhaar numbers and can still track/profile users.

This shortcoming is identified by Shweta Agarwal et al in [12] but is not addressed adequately. Their solution is to maintain a unidirectional reverse linking from local ids to global Aadhaar numbers. If this linking is stored with service providers, this solution is no better than UIDAI's as service providers still have access to Aadhaar numbers. They refer to a solution provided in [2] which involves generating multiple cryptographically embedded local ids, and storing

these in the smart cards of the users. This is not feasible with Aadhaar system as Aadhaar users do not have smart cards. We propose a solution based on Crypto-book [8] which solves the problem of preventing profiling and correlation across multiple applications for applications using a single sign-on. Note that the assumptions made for in the Crypto-book solution are reasonable (for example, the feasibility of a secure channel) and we assume the same. A brief summary of the Crypto-book protocol is provided in the appendix.

In single sign-on system, an online service like Facebook or Google provide identities to users. Users can use these identities for accessing third party applications. The same problem arises here: multiple third party applications can correlate their data and profile/track users. The solution proposed in Crypto-book[8] is for third party applications not get the actual identities of the users but get pseudonyms. They achieve this by adding two more layers between the identity provider and third party application (multiple credential producers and credential consumers) and the use of cryptographic blind signatures. Credential producers generate cryptographic, unlinkable but accountable credentials. Credential consumers use these cryptographic credentials to create pseudonyms which are then provided to the third party application. These credential producers and consumers are servers run by unbiased anonymous authorities who ideally should be independent of UIDAI; if collocated there can be possibilities of collusion.

Crypto-book proposes a solution referred as "at-large" which allows a user to share limited information with third party applications using blind signatures. An instance of "at-large" provides a solution to our problem: this can be used to hide Aadhaar number and other unique data and to provide only nonunique data like name, address etc. The client mentioned in the following discussion is the device that performs authentication which is controlled by a service provider (AUA). In Crypto-book, Application-embedded consumer is a type of a credential consumer which is embedded in application itself. We choose this type of credential consumer[8].

Application of Crypto-book Crypto-book works in a web environment but this is not the case with Aadhaar. We need a different access method for Aadhaar while ensuring that the authentication process should be initiated by the user. We apply Crypto-book algorithm for eKYC API as securing eKYC API also implies securing YES/NO mode because eKYC API gives out more information. The description of actors in our system are

- User: The citizen of India who wants to avail services from AUA (service provider). Citizens use Aadhaar infrastructure to verify their identity to AUAs
- Client: Client is the device which initiates the verification of identity of user.
- Credential Producer: Is the entity which allows an user to be verified to the AUA without disclosing Aadhaar number to Client.

We propose the use of mobile phones for initiating the authentication process with the authentication flow as follows:

An user will have to follow following steps for Aadhaar authentication:

1. The user communicates the Aadhaar number to a cluster of Credential Producers via an independent channel like SMS or a call to IVRS. This step opens a request and an identifier $tempId$ is generated randomly and returned to the user. This identifier is temporary and can be reused after a certain period of time.
2. The user provides $tempId$ and all the required information along with a fingerprint or iris scan to the client.

After the user completes his part, the steps involved in the identity verification are as follows:

1. The client generates a random number $localID$. This $localID$ is the local identifier of a user for that client.
2. The client sends
   - a hash $h = H(localID, clientID)$ where $localID$ is a random number and $clientID$ is a identifier for client.
   - a biometric scan.
   - the demographic information that client wants to be verified
   - $tempId$

   to k different credential producers.
3. The Credential Producers find Aadhaar number previously received using $tempId$ and get the data corresponding to the Aadhaar number using the biometric information and eKYC API.
4. If demographic information is verified, the Credential Producer p_i signs h to produce blind signature s_{p_i}. Along with signature, it sends the non-unique data from the aadhaar data that it fetched from CIDR to the client.
5. The client verifies the signatures sent by producers. If at least one of the signature is verified, client considers the authentication to be successful and accepts the data sent by the credential producer.

   Figure 2 gives the summary of our adaptation of Crypto-book for Aadhaar.

3.4 Proof of Privacy of Aadhaar Number

Client Side: The client has access to following data only

1. $localID$, generated by client itself
2. Biometric data,
3. Demographic Information
4. $tempId$

- The client cannot infer Aadhaar number from $tempId$ because that mapping is only present with Credential Producer.
- Other two data items are unrelated to Aadhaar Number.

Therefore, client cannot infer Aadhaar number of the user. Also, $localID$ is considered to be the local identifier for the particular user in the given AUA. This number is randomly chosen by every AUA during authentication of each user. Therefore every AUA will have different local identifier for a particular user. Hence there is no chance of profiling a user across multiple clients.

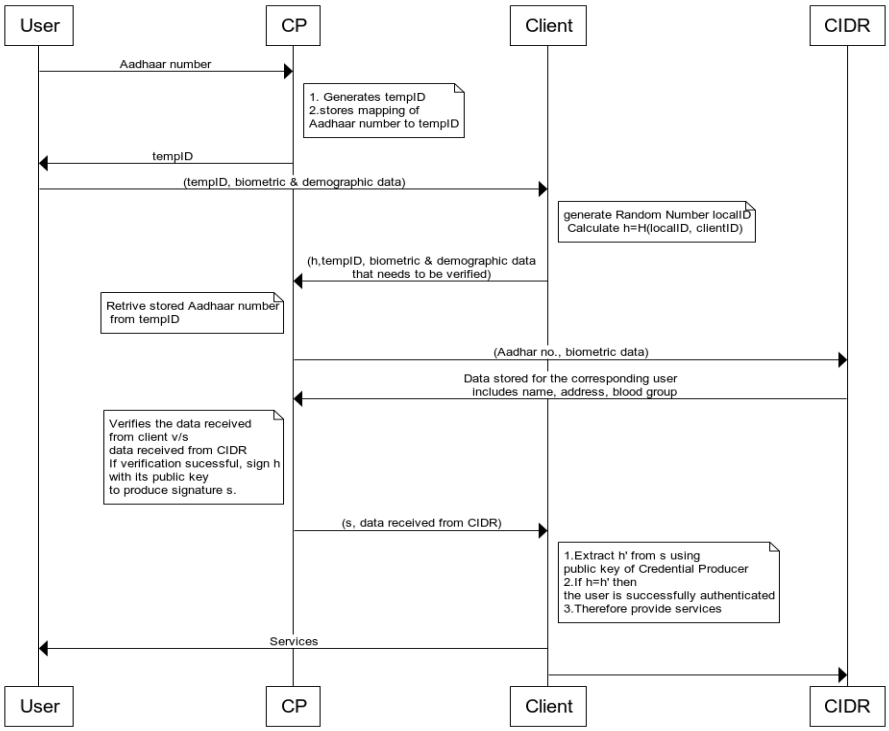

Fig. 2. Summary of Authentication Sequence

Legend

1. **tempID**: Temporary Identifier randomly generated by Credential Producer for a request generated by the User
2. **Biometric data**: This is the biometric data provided by User.
3. **Demographic data**: This data is the data that the clients needs to verify before providing services.
4. **clientID**: This is identification of client.
5. **h**: is the hash generated with inputs clientID and localID.

Credential Producer: The credential producer only verifies the biometric and demographic data and signs the message m only if it verifies the data. Credential Producer does not have access to $localID$. It signs h blindly without knowing $localID$. This signifies that the Credential Producer cannot infer the client who is trying to authenticate the given user and the $localID$ used by that client. Hence Credential Producer cannot profile the user or track the user from use of Aadhaar number.

Further, we prove the privacy of Aadhaar number using ProVerif protocol verification tool [4]. Our setup is as follows.

- We setup 3 processes named **User** representing the user, **Client** representing the client in our protocol and **CP** representing the credential producer.
- We declare 2 channels: **Internet** which is vulnerable to an external attacker and **Cellular**, an encrypted private channel between User and CP. This channel is used when the client registers a request with Credential Producer.

We prove the privacy of the Aadhaar number as follows

1. We run the whole protocol 2 times with different Aadhaar numbers.
2. ProVerif proves that these two runs are observationally equivalent with respect to an external attacker.
3. Client has only 1 extra piece of data compared to attacker which is $localID$. This $localID$ is generated by client itself.
4. Therefore, the two runs of the protocol are observationally equivalent to the client also.
5. Hence the client cannot infer the Aadhaar number under this protocol.

We have uploaded our ProVerif Files at [10].

3.5 Proof of Correctness of Authentication

We prove correctness of authentication using ProVerif. ProVerif provides a notion of **events** and **queries**. The **events** track various actions taken in the protocol while **queries** can be used to prove certain properties. One use of queries is to prove that if an event e_1 has occurred then it implies that some event e_2 has occurred.
Our setup is as follows

- We setup 3 processes named **User** representing the user, **Client** representing the client in our protocol and **CP** representing the credential producer.
- We declare 2 channels, **Internet** and **Cellular**; we assume the channels are secured with end to end encryption and use of digital signatures to avoid masquerading attacks.
- We declare 4 events
  1. $clientConsidersVerified(tempID)$: This event is recorded when client considers some user who provided $tempID$ verified.

2. $cpActuallyVerified(tempID, data)$: This event is recorded when Credential Producer actually verifies the $data$ corresponding to Aadhaar number corresponding to $tempID$
3. $cpGeneratedId(Aadhaar\_number, tempID)$: This event is recorded when Credential Producer generates a $tempID$ for a particular Aadhaar number
4. $dataSubmittedToCP(Aadhaar\_number)$: This event is recorded when a user submits a particular Aadhaar Number to a Credential Producer

With help of ProVerif we prove that following correspondence of events hold.

1. If an event $clientConsidersVerified(tempID)$ occurred, then it implies event $cpActuallyVerified(tempID, data)$ occurred
2. If event $cpActuallyVerified(tempID, data)$ has occurred, then it implies event $cpGeneratedId(tempID, Aadhaar\_number)$ occurred.
3. If event $cpGeneratedId(Aadhaar\_number, tempID)$ occurred, then it implies event $dataSubmittedToCP(Aadhaar\_number)$ occurred.

Note that the arguments to the events ensure that the events correspond to only one set of $Aadhaar\_number$ $data$ and $tempId$. This correspondence implies that when a person is verified by the protocol, the person is actually who he claims to be.

3.6 Other Considerations

We assume that all users have a mobile phone. A case when the user does not have a mobile phone can be handled with the web interface at service provider and authentication by the user using passwords. This solution solves our problem because the process is initiated by the user and not by the AUA. Thus AUA never gets to see the Aadhaar number of the user and hence cannot correlate the number with other service providers. Also, as stated before, $localID$ becomes a local identifier for the user at that service provider. The credential producer does not have access to $localID$. This scheme also provides accountability in case of abuse by a user. The credential producers should maintain a mapping of Aadhaar numbers to h. This can be used by authorities to identify user's Aadhaar number from the local identifier.

Our integrated solution handles all the above in a single process. For example, the identification vs authentication issue is moot as we require user initiated authentication as well as a live "OTP"-based solution. The surveillance issue is handled through a system that uses blind signatures.

For this new system to work, some existing practices need to be abolished such as

1. Collection of copies of Aadhaar id cards should be stopped as it gives service providers access to human readable Aadhaar number
2. Awareness about proper use of Aadhaar number should be disseminated among citizens; specifically, citizens should not provide their Aadhaar numbers to service providers.

As the Aadhaar systems is already operational, there is indeed a significant problem as of now. However, the problem is still not that serious as it has been in light use for just a few years with the expected heavy use only in the immediate future. Hence, using Aadhaar system itself, it may be possible to assign safely a new Aadhaar number for each citizen and use the new Aadhaar number in the future but with privacy guarantees. Maybe one can even advise users to use the old Aadhaar for certain applications such as for emergency/disaster situations where informed consent etc are not really meaningful and use the new ones for almost all else.

Anonymous data for analytics Data analytics is a powerful tool. But there is a risk of unauthorized correlation of data while applying data analytics on Aadhaar data. The risk factors can be removed from the system by using credentials producers and credential consumers. Credential consumers can generate suitably anonymous data for data analytics applications. The system just has to make sure that the application uses pseudonyms generated by the credential consumer.

3.7 Systems Design Architecture Issues

One can view the Aadhaar system also as a concentric set of layers; the innermost being the core UIDAI functionality of providing Y/N responses ("microkernel"), a middle layer that is the backend of applications that use UIDAI ("kernel") and an outermost layer for the front end of the applications ("apps"). If such is the design, going by core software engineering principles, the question to be posed is what kinds of data can be safely exchanged across these layers.

If raw biometric data is sent across to the innermost layer, an important question is whether we can "validate the inputs"? As biometric data can be complex data objects with a big application suite (50MB is not uncommon) required for its use, any zero-day attacks in the suite can be problematic. Note that malware in jpeg or other image files has been used to get access by attackers. Hence, processing the biometric data in a less critical area is advisable but this means that only a (text) summary is provided to the innermost core. However, replay attacks/DoS may now be possible as only text needs to be generated! Hence, we need again a liveness detection system.

4 Access Control Models

Since the Aadhaar system is a complex one with many actors, the internal processes in UIDAI may be required to follow an access control architecture; we provide such a solution below modeling Aadhaar using a Trust and Role Based Access Control Model (TDBAC). While we also need to ensure the integrity of the attributes used in the access control model, which can be implemented using PKI-like system, we do not discuss it in this paper due to reasons of space.

Furthermore, there are many access control requirements on a system that guarantees security and privacy; this requires a careful process based approach but we take only a few examples.

4.1 Trust and Role Based Access Control Model

Trust and Role Based Access Control Model (TDBAC)[6] is an extension of traditional role-based access control model that uses attributes to determine trust. The roles and the attributes are together called factors of authorization. The factors of authentication decide whether a particular request is to be allowed or not. Many factors don't change frequently and these factors can be embedded in the permissions assigned to roles. But many factors like type of request, time of request, environmental factors, authorization from another entity, special tokens could also be considered to decide on the authorization of a request. Attributes present different conditions under which the roles are granted certain permissions. The values of these attribute, subject to constraints specified at design of the system, also account in successful validation. The attributes give a finer and more constrained role based access control; properly chosen set of attributes also give an excellent way to argue about the information flow of the data.

To summarize in our model,

$$U \rightarrow R \rightarrow A \rightarrow P$$

Where U is the user, which maps to roles (R), which maps to relevant attribute checks(A), on passing which the access request is allowed or disallowed. A user creates a session with the required resources and activates one or more roles at any given time in the given session.

5 Instantiation of Access Control Model

5.1 Set of roles (R)

R = {CIDR-Operator, CIDR-SysAd, CIDR-Service, ASA-Operator, ASA-Service, AUA-POS, AUA-Service, AUA-SysAd, User, Govt-Agency-X, Enrollment-Agency, Auditor}

5.2 Description of Roles

1. **CIDR-Operator:** Employee working at CIDR data center with access to the systems at data center.
2. **CIDR-SysAd:** CIDR employee who sets the rights of people working at CIDR.
3. **CIDR-Service:** The actual process which serves authentication services.
4. **ASA-Operator:** The employee working at ASA, who has access to computers at ASA.
5. **ASA-Service:** The process that handles the authentication requests.

6. **AUA-POS:** The Point of Sale device used by the AUA.
7. **AUA-Service:** The AUA process that sends out the authentication requests. D
8. **User :** is the end user who avails the identification services provided by UIDAI to get the services from the AUAs
9. **Govt-Agency-X :** is the government agency X that requires Aadhaar data for various reasons like formulation of welfare schemes etc.
10. **Enrollment-Agency:** The agency appointed by government to collect data from citizens.
11. **Auditor:** is the third party who acts as key holder for various encryption keys and also which is responsible for authorization of various internal audits, inspections, and health of the programs running on the CIDR servers

5.3 Attributes

The set of attributes A is defined as follows.
A = {AUA-ASA-Auth-Token, IP address, Request-Time, Device-Auth-Token, Warrant-X-Y, Disaster-X, Inspection, User-Token Sanctioned-Z}

5.4 Description of attributes

Obvious ones elided.

1. **AUA-ASA-Auth-Token:** The authentication token should be obtained during the initial authentication of AUA to CIDR
2. **Device-Auth-Token:** The authentication token that the POS device is legitimate and verifiable; use of trusted devices is one possibility.
3. **Warrant-X-Y:** Warrant issued by the organization X for person Y. This attribute is a token like one issued in Kerberos. It should also be signed by the organization X.
4. **Disaster-X:** This attribute is present when the request is for disclosing the data object X from the database in case of the disaster/emergency issued to government agency X. This token should be signed by the issuing organization.
5. **Inspection:** This attribute denotes that the request is for the inspection. This attribute should be signed by the auditor, which is the third party mentioned in the paper.
6. **User-Token:** This attribute is the user token submitted for authorization. Its values can be Demographic, Bio metric-Fingerprint, Bio-metric-Iris, OTP. This is to ensure that authentication is never attempted without consent. One way to implement this is to use the Crypto-book ("zero knowledge") authentication between AUA and CIDR as discussed earlier.
7. **Sanctioned-Z:** This is the attribute when an analysis task of the Govt-Agency-Z is sanctioned by the Auditor.

5.5 Permissions

Data Objects To identify the permissions required for this model we identified the data objects which are present in the interactions. The set of data objects D = {Req, Reply, Name, Address, DOB, Nationality, Biometric-Fingerprint, Biometric-Iris, Blood-Group, Aadhaar-Number, Phone-Number, Email, Aadhaar-Local-Linking, Logs, CIDR-UR-Assignment, ASA-UR-Assignment}

Description of Data Objects Obvious ones elided.

1. **Req:** This is the request received at CIDR. Examples of this request are identification request from AUA or data request from various govt agencies.
2. **Reply:** This is the reply that the CIDR responds to the request. It is Yes/No for authorization request or data for the data request from various agencies.
3. **Aadhaar-Local-Linking:** This is mapping between Aadhaar number and local identifier used by different organizations as recommended by UIDAI
4. **Logs:** These are the access logs.
5. **CIDR-UR-Assignment:** The assignment of users to roles at CIDR
6. **ASA-UR-Assignment:** The assignment of users to roles at ASA

Operations The authorizations are to be granted on the operations to be performed on the above objects; these are the set O = {Add, Read, Modify, Store, Read-Anonymity, }. Permissions are given by the set P defined as follows:

$$P = D \times O$$

Access Control Checks As an example, we give 4 rules that need to be checked out of the many (for example, at least 17 such "rules" can be picked out from the English descriptions in Shweta Agarwal's paper [12]) but due to lack of space we do not provide further details):

1. **Rule:** Identification/Authentication of individuals using the global Aadhaar number should not be possible without consent.
   This rule can be enforced by following encoding of roles, attributes and permissions. For all roles for performing read operations to Aadhaar number of a particular user, User token attribute should be present and valid.
2. **Rule:** Employees of UIDAI should not be able to access the user data.
   For role of CIDR-SysAd, read permission should be absent for all user data under all relevant tokens to role of CIDR-SysAd.
3. **Rule:** Identification and authentication should not be possible without consent using public data such as demographic and biometric data.
   Read permission for any role on demographic data should only be granted if user token is present and Valid
4. **Rule:** Co relation of Aadhaar Number should not be possible
   This rule can be enforced by not allowing read permissions on Aadhaar number data object for role of AUA.

6 Conclusions

In this paper, we have pointed out some flaws in the current Aadhaar system design such as the lack of distinction between authentication and identification which leads to confusions and insecure design. Also, some important loopholes in security exist such as lack of liveness detection through which it is possible to attack the system using replayed biometrics. These attacks can be used for identity thefts to avail services for e.g. acquiring a telecom service on a stolen identity. It is possible to profile or track individuals in the current system whereas in a citizen centric architecture identification and authentication without users consent must never be possible. This, in turn, solves the problem of illegitimate tracking and profiling. We have provided solutions to each of these problems but these need to be thoroughly evaluated in a realistic context. Furthermore, the internal processes in UIDAI may be required to follow an access control architecture; we have provided such a solution modeling Aadhaar as TDBAC.

A Crypto-book Algorithm

The architecture of Crypto-book is shown in Figure 3. The description of the 4 entities in Crypto-book are as follows:

1. **The federated identities producer(F):** like UIDAI, Facebook, Google who provide identities and provide single sign on service
2. **Credential Producer(CP):** who verifies federated identities and provides partially blind credentials to consumer
3. **Credential Consumer (CC):** who takes as input the credentials produced by credential producer to produce pseudonyms that are presented to third party applications
4. **Third party applications (A):** These applications use the identities provided by **F**, after authentication by the user at interface provided by **F**

Crypto-book uses blind signatures to produce pseudonyms which are presented to third party applications. Blind signatures are cryptographic primitive in which a requester can request a signer to sign a message where signer does not learn the content of the signed message. For blind signature, requester first obscures the message m with some secret to produce m' which is then signed by the signer to produce blinded signature s'; Because requester knows the secret he can remove the blinding factor and send m and unblinded signature s to the receiver. A verifier can then verify the signature using public key of the signer[7]. The following are important steps in Crypto-book; here, the client is the user, credential producer is the signer and credential consumer is the verifier.

A.1 Producing credentials

To obtain a t "at-large" credential for use with consumer with identity idc, a client first generates a random value r which identifies the credential. The

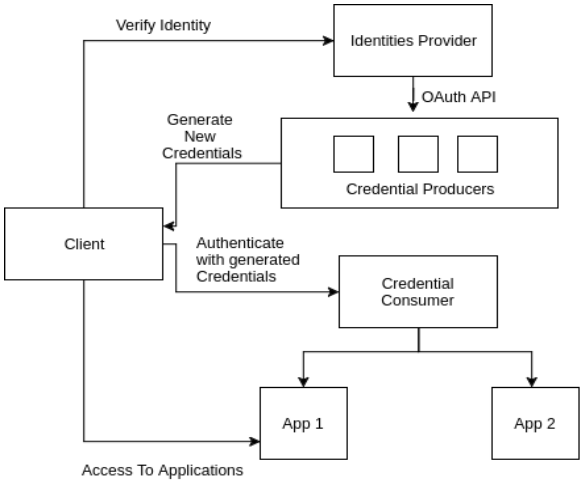

Fig. 3. Architecture of Crypto-book
places in

client hashes this value r with the identity of the consumer to produce message $m = H(r, idc)$. The client then contacts at least t of the n credential producers with signature requests, uniquely blinding the message m to produce m' for each request. Before signing the message, each credential producer verifies the client's federated identity and, if successful, returns blinded signature s'_i to the client. The client unblinds the signatures from each of the credential producers to obtain a vector of unblinded signatures $s_1, s_2, ...s_t$ which serves at the at-large credential for anonymous identity r with credential consumer c.

A.2 Consuming credentials

To authenticate with a credential consumer requiring a threshold t at-large credential, a client must provide the credential consumer with the value r defining their anonymous identity along with a vector $s_1, s_2, ...s_t$ of signatures from at least t unique credential producers. The consumer first hashes this value with its own identity to produce message $m = H(r, idc)$. The consumer, using the public keys of the credential producers, then verifies that each signature is, in fact, valid for message m and, if successful, authenticates the client as anonymous identity r.

References

1. Amber Sinha, S.K.: Information security practices of aadhaar (or lack thereof). Tech. rep., The Center for Internet and society, `https://cis-india.org/internet-governance/`

information-security-practices-of-aadhaar-or-lack-thereof/at_download/file
2. Angell, I.: The identity project an assessment of the uk identity cards bill and its implications. Tech. rep., London School of Economics (2005), http://www.lse.ac.uk/management/research/identityproject/identityreport.pdf
3. BBC: Turkish authorities 'probing huge id data leak (2016), http://www.bbc.com/news/technology-35978216
4. Blanchet, B., Cheval, V., Allamigeon, X., Smyth, B.: Proverif: Cryptographic protocol verifier in the formal model (2010), http://prosecco.gforge.inria.fr/personal/bblanche/proverif
5. GoI: The aadhaar (targeted delivery of financial and other subsidies, benefits and services) act, 2016. Act in Govt of India, by MINISTRY OF LAW AND JUSTICE (2016)
6. Jemel, M., Azzouna, N.B., Ghedira, K.: Towards a dynamic access control model for e-government web services. In: Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific. pp. 433–440. IEEE (2010)
7. Maheswaran, J.: Building Privacy-Preserving Cryptographic Credentials from Federated Online Identities. Ph.D. thesis, Yale University (2015)
8. Maheswaran, J., Jackowitz, D., Zhai, E., Wolinsky, D.I., Ford, B.: Building privacy-preserving cryptographic credentials from federated online identities. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (2016)
9. Meyer, R.: Long-range iris scanning is here (2015), https://www.theatlantic.com/technology/archive/2015/05/long-range-iris-scanning-is-here/393065/
10. Rajput, A., K.Gopinath: Proverif files (2017), https://github.com/the-elves/ICISS-codes
11. Sharma, A.: Direct benefit transfer leads to rs 50,000-crore savings for government in 3 years (2016), http://economictimes.indiatimes.com/news/economy/finance/direct-benefit-transfer-leads-to-rs-50000-crore-savings-for-government-in-3-years/articleshow/57240387.cms, the Economic Times
12. Shweta Agrawal, Subhashis Banerjee, S.S.: Privacy and security of aadhaar: A computer science perspective, http://www.cse.iitm.ac.in/~shwetaag/papers/aadhaar.pdf
13. The-Economic-Times: Uidai lodges fir against axis bank and two more firms for tampering with aadhaar biometrics (2017), http://economictimes.indiatimes.com/articleshow/57325951.cms
14. The-Guardian: India goes from village to village to compile worlds biggest id database (2016), https://www.theguardian.com/world/2016/jun/28/india-village-compile-worlds-biggest-id-database-aadhaar
15. UIDAI: (2016), http://www.licindia.in/getattachment/Bottom-Links/Tenders/RFP-for-Two-Factor-Authentication-and-Aadhaar-enab/STQC-UIDAI-BDCS-03-08-UIDAI-Biometric-Device-Specifications-_Authentication_1.pdf.aspx
16. UIDAI: The rule of thumb in identity (2016), https://uidai.gov.in/images/news/rule_of_Thumb_in_identity_13042017.pdf
17. Zahid Akhtar, Christian Micheloni, G.L.F.: Biometric liveness detection: Challenges and research opportunities. IEEE Security and Privacy, 13(5) (2015)