Solution to Aadhaar Linking Problem
Why Crypto-Book won’t work for Aadhaar?

• Aadhaar does not have web interface like single sign on
  • The client device is not owned by user

• Aadhaar solution needs to be independently designed
  • Crypto book framework useful for deeper problems

• We want user initiated process

• Our Solution...
Authentication Sequence

1. Generates tempID
2. Stores mapping of Aadhaar number to tempID

(tempID, biometric & demographic data)

Retrive stored Aadhaar number from tempID

Verifies the data received from client v/s data received from CIDR. If verification successful, sign h to produce signature s.

(h, tempID, biometric & demographic data that needs to be verified)

Generate Random Number localID
Calculate h[H(localID, clientID)]

(Aadhar no., biometric data)

Data stored for the corresponding user includes name, address, blood group

1. Extract h’ from s using public key of Credential Producer
2. If h=h’ then the user is successfully authenticated
3. Therefore provide services
Setup
Citizen

- Anyone who wants to get authenticated
Service Provider
And his device

• Examples like banks, telecoms etc,

• Client device present with user in crypto book, which is not a good assumption here
CIDR

• Central Store of Identities
Credential Producer

- Anonymization Service
- Multiple credential producers are there but here let's consider only one
Color Coding

- The colors represent different data
- Information with each party in protocol is shown below that party

Aadhaar Number: xyz

Indian Institute of Science
Protocol
Phase 1: User-CP Interaction (Request Initiation)

Step 1

User -> Credential Producer Front end
- Via SMS/ivrs
- Info sent
  - Aadhaar number

Aadhaar Number: xyz

CIDR

Credential Producers

Client

Service Provider
Phase 1: User-CP Interaction (Request Initiation)

Step 1

User -> Credential Producer Front end
- Via SMS/ivr
- Info sent
  - Aadhaar number

Aadhaar Number: xyz

CIDR

Service Provider

Client

Indian Institute of Science
Phase1 : User-CP Interaction (Request Initiation)
Step 2

• Propagate the request to all the CPs
• At each Credential Producer:
  • Credential Producer -> User
    • CP creates a request and assigns a reusable identifier to that request say tempID1
    • Return tempID1 to user

Aadhaar No (xyz)
TempID (tempID1)
Info to verify + biometrics
LocalID (localID1)
Phase 1: User-CP Interaction (Request Initiation)

Step 2

- Propagate the request to all the CPs
- At each Credential Producer:
  - CP creates a request and assigns a reusable identifier to that request say tempID1
  - Return tempID1 to user
Phase 1: User-CP Interaction (Request Initiation)

Step 2

• Propagate the request to all the CPs
• Multiple CPs
• At each Credential Producer:
  • Credential Producer -> User
    • CP creates a request and assigns a reusable identifier to that request say tempID1
    • Return tempID1 to user

- Aadhaar No (xyz)
- TempID (tempID1)
- Info to verify + biometrics
- LocalID (localID1)
Phase 2: User-Client Interaction (Request Forwarding)

Step 1

- User provides client the required information and the tempID1

- Aadhaar No (xyz)
- TempID (tempID1)
- Info to verify + biometrics
- LocalID (localID1)
Phase 2: User-Client Interaction (Request Forwarding)
Step 1

- User provides client the required information and the tempID1
Phase 2: User-Client Interaction (Request Forwarding)
Step 1

- User provides client the required information and the tempID1
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - M=H(id_c, localID)
  - Data to be verified
  - tempID
- These messages are sent to t CPs

**CIDR**

**Credential Producers**

**Generate random #**

**Indian Institute of Science**
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - $M = H(id_c, localID)$
  - Data to be verified
  - tempID
- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)
Step 1

- Client -> CP
  - M=H(id_c, localID)
  - Data to be verified
  - tempID
- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - M = H(id_c, localID)
  - Data to be verified
  - tempID

- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - $M = H(id_c, \text{localID})$
  - Data to be verified
  - tempID

- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - $M = H(id_c, localID)$
  - Data to be verified
  - tempID

- These messages are sent to $t$ CPs
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - M=H(id_c, localID)
  - Data to be verified
  - tempID
- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)
Step 1

- Client -> CP
  - M=H(id_c, localID)
  - Data to be verified
  - tempID
- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)
Step 1

- Client -> CP
  - \( M = H(id_c, localID) \)
  - Data to be verified
  - tempID
- These messages are sent to t CPs
Phase 3: Client-CP Interaction (Data ver)

Step 1

- Client -> CP
  - M=H(id_c, localID)
  - Data to be verified
  - tempID
- These messages are sent to t CPs
Phase 4: CP-CIDR Interaction (Data ver)

Step 2

• CP can identify the particular request using tempID1

• It verifies the information through normal UIDAI API

• If UIDAI replies Yes:
  • Sign m and send back signature

• Else
  • Send an error message

• Note that CP does not know what localID1 is since it is hashed.
Phase 4: CP-CIDR Interaction (Data ver)
Step 2

- CP can identify the particular request using tempID
- It verifies the information through normal UIDAI API
- If UIDAI replies Yes:
  - Sign m and send back signature
- Else
  - Send an error message

- Note that CP does not know what localID1 is since it is hashed.
Phase 4: CP-CIDR Interaction (Data ver)

Step 2

- CP can identify the particular request using tempID
- It verifies the information through normal UIDAI API
- If UIDAI replies Yes:
  - Sign m and send back signature
- Else
  - Send an error message
- Note that CP does not know what localID1 is since it is hashed.
Phase 4: CIDR-CP Interaction (Data ver)

Step 2

- CP can identify the particular request using tempID.
- It verifies the information through normal UIDAI API.
- If UIDAI replies Yes:
  - Sign m and send back signature.
- Else:
  - Send an error message.
- Note that CP does not know what localID1 is since it is hashed.
Phase 4: CIDR-CP Interaction (Data ver)

Step 2

- CP can identify the particular request using tempID
- It verifies the information through normal UIDAI API
- If UIDAI replies Yes:
  - Sign $m$ and send back signature
- Else
  - Send an error message
- Note that CP does not know what $localID1$ is since it is hashed.
Phase 4: CIDR-CP Interaction (Data ver)
Step 2

- CP can identify the particular request using tempID
- It verifies the information through normal UIDAI API
- If UIDAI replies Yes:
  - Sign m and send back signature
- Else
  - Send an error message
- Note that CP does not know what localID1 is since it is hashed.
Phase 4: CIDR-CP
Interaction (Data ver)
Step 2

• CP can identify the particular request using tempID

• It verifies the information through normal UIDAI API

• If UIDAI replies Yes:
  • Sign m and send back signature

• Else
  • Send an error message

• Note that CP does not know what localID1 is since it is hashed.

Indian Institute of Science
Phase 4: CIDR-CP Interaction (Data ver)

**Step 3**

- CP can identify the particular request using tempID
- It verifies the information through normal UIDAI API
- If UIDAI replies Yes:
  - Sign m and send back signature
- Else
  - Send an error message
- Note that CP does not know what localID1 is since it is hashed.
Phase 4: CIDR-CP Interaction (Data ver)

Step 3

- CP can identify the particular request using tempID
- It verifies the information through normal UIDAI API
- If UIDAI replies Yes:
  - Sign m and send back signature
- Else
  - Send an error message
- Note that CP does not know what localID1 is since it is hashed.
Phase 5: CP-Client Interaction (Sig ver)

Note that CP does not know what localID1 is since it is hashed.
Phase 5: CP-Client Interaction (Sig ver)

Note that CP does not know what localID1 is since it is hashed.
End

Client verifies the signature and uses localID1 as local identifier

If signed, verified. And localID1 is identifier for that person

Aadhaar No (xyz)
TempID (tempID1)
Info to verify + biometrics
LocalID (localID1)

Services

Indian Institute of Science
Pro-verif

• Assumes cryptographic entities to be black box
  • Dolev-Yao model

• Constructors, destructors and rewrite rules
  • fun \texttt{enc} ( \texttt{bitstring} , \texttt{key} ): \texttt{bitstring}.
  • reduc forall \( m: \texttt{bitstring} , k: \texttt{key} \); \texttt{dec} ( \texttt{enc} (m,k),k) = m.

• Primitives and Processes

• Events
Pro-verif

- Assumes cryptographic entities to be black box
  - Dolev-Yao model
- Constructors, destructors and rewrite rules
  - fun \texttt{senc}(\texttt{bitstring}, \texttt{key}) : \texttt{bitstring}.
    - reduc forall \texttt{m} : \texttt{bitstring}, \texttt{k} : \texttt{key} ; \texttt{sdec}(\texttt{senc}(\texttt{m}, \texttt{k}), \texttt{k}) = \texttt{m}.
- Primitives and Processes
- Events
Pro-verif

• Assumes cryptographic entities to be black box
  • Dolev-Yao model

• Constructors, destructors and rewrite rules
  • fun \texttt{senc (bitstring, key): bitstring}.
    • reduc forall m: bitstring, k: key; sdec (senc (m, k), k) = m.

• Processes, Private and Public

• Attacker

• Events
Proof of secrecy of Aadhaar number from AUA

- Analogous to proofs based on Non Interference
- Proverif Observational Equivalence
- Modelling one of the party of the protocol as malicious

Steps of proof
- Run protocol twice with different Aadhar numbers
- Proverif proves observational equivalence between the processes
- Observational equivalence implies that attacker cannot know uid
- Comparing Attacker and AUA
Phase 3: Client-CP Interaction (Data verification)

Step 1

- **Client** \(\rightarrow\) **CP**
  - \(M = H(id_c, localID)\)
  - Data to be verified
  - tempID

- These messages are sent to t CPs
Proof of correctness of Authentication

• Pro-verif events

• How is authentication guaranteed in current setting?
  • You have submitted the correct data to AUA for verification
  • AUA considers you authenticated when CIDR responds YES
  • CIDR will respond yes if
    • You have enrolled your correct data and Aadhaar has verified it before adding to CIDR

• This is one to one correspondence between events

• We also proved that this correspondence exists in our protocol with help of Proverif