Contents

1 Authentication **2**

1.1 Players: . . . 2

1.1.1 User: A single identity of a user. User wants to authenticate with this identity at Service. . . . 2

1.1.2 Service: A service that has many users. Users can register/login iff they are able to authenticate here. . . . 2

1.2 Procedure: . . . 2

2 Recurring ('Automatic') Authentication **3**

2.1 Players: . . . 3

2.1.1 User . . . 3

2.1.2 Service . . . 3

2.2 Procedure: . . . 3

3 Verification **4**

3.1 Players: . . . 4

3.1.1 User . . . 4

3.1.2 Verifier (Service) . . . 4

3.2 Procedure: . . . 4

4 Data Types: **4**

4.1 List of 'To Be Verified Datasnippets'(TBVDs): . . . 4

4.1.1 Each element of list is (unordered) set of alternatives. 4

4.1.2 Each alternative is a $\{\text{key, verified}_{\text{id}}\}, where `key` issome(conventionalized)stringvalue,$

4.1.3 A special alternative is called 'none', which is used in case of optional datasnippets. . . . 4

4.2 Verified Datasnippets (VDs): . . . 4

4.2.1 Each Datasnippet is: $\{\text{subject}_{\text{id}}, key, verifier_{\text{id}}, data, revocation_{\text{ref}}, verifier_{\text{signature}}\}$

4.2.2 Validity can be verified by Service by checking if the status of the Revocation Contract is modified to be 'revoked'. . . . 5

4.3 Revocation Token . . . 5

4.3.1 Stored in Blockchain by Service (at the same time that a Verified Datasnippet is provided to a User). . . . 5

4.3.2 Contains a list of addresses of Service Contracts or Identity Contracts that are allowed to revoke it at a later time (which might only be Service, but possibly also others). . . . 5

4.3.3 Contains a field that can be changed which is 'valid' at first, and might be changed (once, irreversibly) to 'revoked'. 5
4.3.4 Is referenced from a Verified Datasnippet. 5
4.4 Authentication Token 5
4.4.1 Encrypted by Service with public key of Service 5
4.4.2 Contains timestamp of Authentication, and possibly some (small amount of) other metadata. 5
4.4.3 User can decide whether to store this Authentication Token as part of his Identity Contract on the Blockchain, or only store it locally. 5

The EpicAuth protocol has two separate parts:

1. Authentication
2. Verification.

As part of Authentication Sequences, a Service might ask for zero or more pieces of Verified information.

1 Authentication

1.1 Players:

1.1.1 User: A single identity of a user. User wants to authenticate with this identity at Service.

1. Has a public/private key pair.

1.1.2 Service: A service that has many users. Users can register/login iff they are able to authenticate here.

1. Has a public/private key pair.

1.2 Procedure:

1. User is on webpage of Service, and can choose to authenticate using EpicAuth.
2. -> User clicks button.
3. <- Service sends list of To Be Verefied Datasnippets(TBVDs) to be provided by user.

4. Client-side software shows User what TBVDs are requested. User can decide between: a) Cancel. Stops process here. b) User can pick which identity they want to use. (Only ones that contain all required TBVDs are selectable) b') After selecting Identity, for each datasnippet, which alternative could be used.
5. -> User sends list of selected VDs to Service.
6. Service checks if VDs are Valid.

a) Invalid. Returns failure response to User. b) Valid. Continue

1. Service creates Authentication Token.
2. <- Service returns Authentication Token to Client.
3. Client can decide where to store Authentication Token for future use.
   - Locally
   - In the Blockchain (so other devices that share same Identity(privkey) can now also see/use this authentication token.
4. Client has been Authenticated Successfully.

2 Recurring ('Automatic') Authentication

2.1 Players:

2.1.1 User

2.1.2 Service

2.2 Procedure:

1. User visits webpage of Service.
2. Client-side software sees that he has an Authentication Token for this Service (by checking local storage, by checking Blockchain Identity Contract for references to Authentication Tokens)
3. -> Client sends Authentication Token to Service.
4. Service decrypts Authentication Token and checks if not yet expired. a) Expired. Procedure stops here. Return failure. (Redirect to normal Authentication Flow and return TBVDs?) b) Continue.
5. <- Service logs in Client, and returns success.

3 Verification

3.1 Players:

3.1.1 User

3.1.2 Verifier (Service)

3.2 Procedure:

1. Through some external, verification-type-based means, some information might be verified for a User's Identity by the Verifier.
2. At this time, the Verifier creates a Verification Datasnippet.
    - Optionally, the Service might create a Revocation Token (that is referenced from the Verification Datasnippet). This Revocation Token is put on the Blockchain.
3. <- Service sends Verification Token to User.
4. User decides whether to store Verification Token (encrypted!) only locally, or also as part of their Blockchain Identity Contract.

4 Data Types:

note: Not all of these are stored on the blockchain!

4.1 List of 'To Be Verified Datasnippets'(TBVDs):

4.1.1 Each element of list is (unordered) set of alternatives.

4.1.2 Each alternative is a {key, $\text{verified}_{\text{id}}$}, where 'key' is some (conventionalized) string value, and '$\text{verifier}_{\text{id}}$' is a reference to the Service that should have verified it.

4.1.3 A special alternative is called 'none', which is used in case of optional datasnippets.

4.2 Verified Datasnippets (VDs):

4.2.1 Each Datasnippet is: {$\text{subject}_{\text{id}}$, key, $\text{verifier}_{\text{id}}$, data, $\text{revocation}_{\text{ref}}$, $\text{verifier}_{\text{signature}}$}

1. $\text{subject}_{\text{id}}$: ID (Blockchain address) of the Identity to which this Verified Datasnippet belongs.

2. key: Key that matches TBVD key.
3. data: string-data.
4. $\text{verifier}_{\text{id}}$: ID of verifier (matches with TBVD $\text{verifier}_{\text{id}}$)
5. $\text{revocation}_{\text{ref}}$: Reference (blockchain address) to location of Revocation Contract. Might be 'null' in case that 'verifier' decided that was okay.
6. $\text{verifier}_{\text{signature}}$: Cryptographic Signature of delimited concatenation of all of above fields.

4.2.2 Validity can be verified by Service by checking if the status of the Revocation Contract is modified to be 'revoked'.

4.3 Revocation Token

4.3.1 Stored in Blockchain by Service (at the same time that a Verified Datasnippet is provided to a User).

4.3.2 Contains a list of addresses of Service Contracts or Identity Contracts that are allowed to revoke it at a later time (which might only be Service, but possibly also others).

4.3.3 Contains a field that can be changed which is 'valid' at first, and might be changed (once, irreversibly) to 'revoked'.

4.3.4 Is referenced from a Verified Datasnippet.

4.4 Authentication Token

4.4.1 Encrypted by Service with public key of Service

1. So later, only Service can read it again.

4.4.2 Contains timestamp of Authentication, and possibly some (small amount of) other metadata.

4.4.3 User can decide whether to store this Authentication Token as part of his Identity Contract on the Blockchain, or only store it locally.