Human Identity and the Quest to Replace Passwords Continued

Sirvan Almasi
Imperial College London

William J.Knottenbelt
Imperial College London

Abstract

Identification and authentication are vital components of many online services. For authentication, password has remained the most practical solution despite it being a weak form of security; whilst public-key cryptography is more secure, developing a practical system has been an impediment to its adoption. For identification there is a lack of widely adopted protocol that isn't dependent on trusted third parties or a centralised public-key infrastructure. The Fiat–Shamir identification scheme solved this issue but by assuming there was a single identity issuer to begin with, whilst in reality you would have multiple identity issuers. Here we introduce deeID, a mobile phone and blockchain-based system that enables multiple Fiat–Shamir identity issuers for identification. deeID is also considered to be an out-of-band authentication scheme that enables better than password security whilst remaining practical. We achieve a decentralised network of identities that is used for identification and authentication, thus providing a common protocol for organisations to use when communicating with respect to a user (identity holder).

1 Introduction

Identification and authentication are an integral part of our interaction with the internet, yet the technologies underpinning these interactions have remained open to attack because the defensive barriers are easily breachable. Password, which underpins most authentication protocols, is a weak form of security [1]: it's open to user errors [2] (such as re-using them [3], forgetting them [4] and sharing them) and it doesn't scale well.

Identification is closely linked to authentication and on the internet it remains to be an open problem. Identification is widely based on archaic processes which involves our knowledge of personal information such as full name, date of birth, and address. Combining weak password-based authentication schemes and poor identification systems we reach an ecosystem plagued by security and privacy concerns at every corner. Large scale data breaches [5] have demonstrated the ease at which identities and user accounts can be stolen [6] and this has damaged the trust we put into online services.

The concept of treating an online service as an adversary is not a new concept and is demonstrated in the work of Fiat and Shamir [7] through what is now known as the Fiat–Shamir cryptosystem. This cryptosystem is a novel method of identification that allows a pair of users to communicate securely without exchanging public keys and it negates the need for a public-key infrastructure (PKI). Its drawback is that it uses a single trusted entity to issue identities. However, in reality the world is far more complex and one requires multiple identity issuers (such as your bank, passports and birth-certificates).

In this paper we design and implement an out-of-band authentication and identification system, named deeID, using the Fiat–Shamir cryptosystem but extended to allow for multiple identity issuers. We use the comparative framework by Bonneau et al. in [8] to compare our system with password-based authentication schemes.

deeID is a mobile phone and blockchain-based system. We achieve the extension of the Fiat–Shamir cryptosystem by leveraging a blockchain network as a decentralised public-key infrastructure (PKI). Using blockchain as a PKI and as an identity management tool are topics of ongoing research. The likes of Namecoin [9], Blockstack [10] and Certcoin [11] are examples of organisations attempting to recreate DNS, identity and certification services using blockchain. Other examples that have used the blockchain as a PKI include [12] for IoT devices and [13] for identity management.

The complexity of our relationship with digital systems (human-to-machine relationship) and the drawbacks of password have motivated us to envision a password-less relationship with the internet and machines around us. The dynamism of identification (uniquely identifying an entity) and authentication (proving who or what one claims to be) in this human-machine relationship further makes password unfit for purpose. Password-based protocols are a *what we know* method of identification, the other two are methods are: *what we* *posses* and *what we are*. Password has remained the dominant end-user authentication due to its practicality [8] but the ubiquitous mobiles phones [14] [15] is already changing that practicality aspect.

The benefits of the proposed system beyond the existing motivation of replacing password as an authentication tool are numerous. Having a single source of identity in an economy can prevent identity and synthetic identity fraud [16].

Our contributions are as follows.

- *Fiat–Shamir cryptosystem extension.* We propose a new approach to the Fiat–Shamir cryptosystem where an ecosystem can have multiple identity issuers. The cryptosystem is explained in Section 2 with its design into deeID in Section 3.
- *Design of deeID*. The proposed system is a mobile phone and blockchain-based authentication and identification scheme. We design the system with the Fiat–Shamir identification scheme that utilises a blockchain network as a decentralised PKI. Design of deeID is explained in Section 3.
- *Implementation of deeID.* The implementation (see Section 4) constitutes of the following components: mobile phone application, Ethereum smart-contracts, web application and accompanying code. The implementation is available online at `https://github.com/deeId`.
- *Mobile phone and blockchain based authentication scheme*. We compare this new class of authentication scheme through deeID using the Semi-structured Evaluation of User Authentication Schemes Framework [8]. This comparison can be found in Section 5.

2 Background

In this section we will explore the definition of identity and its representation, existing authentication schemes, and identification cryptosystems.

2.1 Identity

The Oxford dictionary defines the word *identity* as *The fact of being who or what a person or thing is*. Identity is a unique representation of an entity. One can uniquely represent a human and their identity by concatenating the person's characteristics, such as: hair colour, genetics, height, and so on. If we gather enough characteristics we can uniquely represent the person with a high probability. This representation is merely a set of characteristics which can form into a very long string. However, such representation is prone to errors, as identification is a process of many steps. Recording such characteristics can be difficult as there can be discrepancies in recording simple characteristics such as eye colour. Then we must think about verifying the identity and the practicality of doing so, too. So, with respect to an identification protocol, we must consider the following:

1. *Representation*. What is being used to represent the entity?
2. *Capture*. How are we capturing this representation?
3. *Recording*. Can it be recorded on a machine?, how is it recorded? and the efficiency of doing so.
4. *Security*. Is it tamper-proof?, is there a risk to impersonation and other relevant threats?
5. *Verification*. Can we verify the identity? is the process error free? and is the process efficient enough (within the context of the application)?

2.2 Authentication Methods

The username/email and password combination has remained the go-to standard for authentication. Other iterations such as password managers, two factor authentication (2FA), and federated systems have made advances in improving its security and usability. Bonneau et al. in [8] provide a unique framework that allows us to compare different authentication schemes. In their paper they provide a unique insight on why password has remained in being a practical system despite its flaws. The comparative framework in [8] compares different web authentication schemes based on 25 factors which are categorised under three headings: usability, deployability and security. As expected most other schemes perform better in security, mix results on usability, however all the other schemes do worse than password on deployability. An overview of their comparative result is visualised in Figure 5 along with the comparative result of deeID.

Federated protocols such as OpenID [17], OAuth [18] and SAML [19] provide a greater level of standardisation, their adoption by tech-giants like Google [20] and Facebook [21] has increased their adoption amongst developers as more users find it more convenient to have have one login credential. Nevertheless, they are still password based. Therefore, as noted above, password is a weak form of authentication and security remains to be a concern. Hardware tokens and phone-based schemes that use special cryptographic keys are significantly more secure than passwords. Though simple categorisation does not mean an automatic better security; implementation remains to be important too. As indicated in the results of [8] (also shown in Figure 5), MP-Auth [22] and IronKey [23] are shown to be less secure than their counterparts in the same category.

Biometric-based authentication schemes are an easy to use schemes. However, the scheme is held back due to technology and the general noise when capturing the data (not deterministic). Biometric schemes generally require a piece of hardware that is only useful for that application and thus makes it impractical for us to carry around all the time. Though, mobile phones have helped by providing face and finger print scanners in recent years. We should also see an increased use of it as mobile phone adoption increases. Given that it is much more difficult to steal biometric data [24] and its ease of use, we should see more of it being used in conjunction with other schemes.

2.3 Identity-based Cryptosystems

The drawbacks of using a public-key cryptosystem are its practicality, certification and trusted party requirements. Such drawbacks are mitigated in identity-based cryptosystems. It essentially allows one to use one's identity (e.g. name, date of birth, place of birth etc.) as a public-key. In such a system everyone would have access to some function f. This function would allow one to compute the public-key from some string which is unique to the individual.

The famed cryptographer Adi Shamir[1] first introduced an identity-based cryptosystem and signature scheme in his 1984 paper *"Identity-based cryptosystems and signature schemes"* [25]. This novel technique is still based on public-key cryptosystems, however, with a *twist* as he describes it himself. Instead of generating and publishing a public-key, the user can use a set of identities and features unique to themselves (e.g. name, date of birth, place of birth etc.) as a proxy for the public-key. The implementation scheme provided by Shamir is only an identity-based signature scheme in this paper.

Shamir describes the scheme ideal for closed groups of users, however, our ecosystem requires an open system, but he also goes on to say it is practical on the national scale.

Key properties described by Shamir in [25]:

- A trusted key generation centre is required to generate user's secret key and issue in the form of smart card.
- Advantage is the ease of use, *"...it can be used effectively even by laymen who know nothing about keys or protocols."*

Security of the system depends on the following:

- Security of the underlying cryptographic functions
- Secrecy of the privileged information at the key generation centres.
- Thoroughness of identity checks at the key generation centres before a smart card is issued.
- Precautions taken by the user to prevent loss, duplication or unauthorised use of their card.

[1]https://www.britannica.com/biography/Adi-Shamir

Shamir only provides a signature implementation scheme for his identity-based system idea. Shamir does return to this idea in subsequent papers in the following years after 1984. In 1986, Fiat and Shamir give us the first concrete solution [7]. They note that *"The new identification scheme is a combination of zero-knowledge interactive proofs [26] and identity-based schemes [25]"*. In this paper [7] they provide both the signature and proof of identity scheme that Shamir proposed in the 1984 paper [25]. Moreover, Shamir along with Feige and Fiat produce a further study on *"Zero-Knowledge Proofs of identity"* [27], this paper discusses and differentiates between zero-knowledge *"proofs of membership"* and *"proofs of knowledge"*.

Feige-Fiat–Shamir Identification Protocol [27] This is the algorithm as described in their paper in 1988 *"Zero Knowledge Proofs of identity"* [27]. We have made some adjustments to the notation to be consistent throughout this paper. Its security is dependent on the intractability of computing the square roots mod n.

Assuming the interaction is occurring between two entities, the *prover A* and the *verifier B*.

A's key generation protocol is as follows:

1. Choose k random numbers $S_1, ..., S_k$ in $\mathbb{Z}_n$
2. Choose each v_j (randomly and independently) as $\pm 1 \cdot (S_j^2)^{-1} \mod n$
3. Publish $v_1, ..., v_k$ and keep $S_1, ..., S_k$ secret

The generation and verification process (i.e. interactions) takes place as follows via a four step process: Repeat t times:

1. A (the prover), picks a random R, and sends $X = \pm R^2 \mod n$
2. B (the verifier), sends a random boolean vector $E_1, ..., E_k$
3. A sends the value $Y = R \cdot \prod_{E_j=1} S_j \mod n$
4. B verifies that $X = \pm Y^2 \cdot \prod_{E_j=1} v_j \mod n$

In their last remarks [27], Feige, Fiat and Shamir note: *"An interesting modification can eliminate the public key directory and lead to a key-less identification scheme"*. We use the identity string to public-key transformation provided in [7] to create public-keys in the system. Implementation of the scheme is given in the Architecture section.

The implementation provided by Fiat and Shamir [7] is for proving ones identity through an interactive manner. However, there had not been any *encryption* schemes until the works of Boneh–Franklin [28] and Cliffor Cock's scheme [29] in 2001.

The systems described above provide an important layer on top of public-key cryptography, public-keys are not the most user-friendly concept, for a national system we require a more friendly scheme that even the laymen can understand and operate. Moreover, the major advantage of the Fiat–Shamir protocol is that it can work with systems that are limited in power and memory.

There are other protocols similar to the Fiat–Shamir protocol, such as the Guillou-Quisquater (GQ) identification scheme [30] (difference is a reduction in the number of messages exchanged and the memory requirements for user secrets). The Schnorr identification protocol [31] depends on the intractability of the discrete logarithm problem unlike the Fiat–Shamir and GQ protocol.

2.3.1 Attacks on Identification Protocols

There are various security threats to identification protocols [32]:

- *Impersonation*. Pretending to be a different person
- *Replay attack*. Use of information from a previously used protocol process, e.g. re-use of a poor signature.
- *Interleaving attack*. Multiple concurrent execution of the protocol and using selective information for possible attacks.
- *Reflection Attack*. Attack on a challenge-response method to essentially tricking the challenger into providing itself with the answer.
- *Forced Delay*. Requires intercepting a message and delaying it until some later time (thus not a replay attack).
- *Chosen-text Attack*. Strategically choosing challenges in an attempt to extract information about the provers secret information.

3 Design

The goal of the proposed system, deeID, is to provide an alternative end-user authentication scheme to password whilst having the ability to provide human identification functionalities. In order to achieve these goals we provide two co-existing functionalities: 1) Identification via Fiat–Shamir cryptosystem (extended via blockchain for multiple identity issuers). 2) Out-of-band (mobile phone-based) authentication using public-key cryptography. Our focus is on authentication and identification for web-based applications but certainly not bounded in capability to this ecosystem. Figure 1 provides an overview of the proposed system.

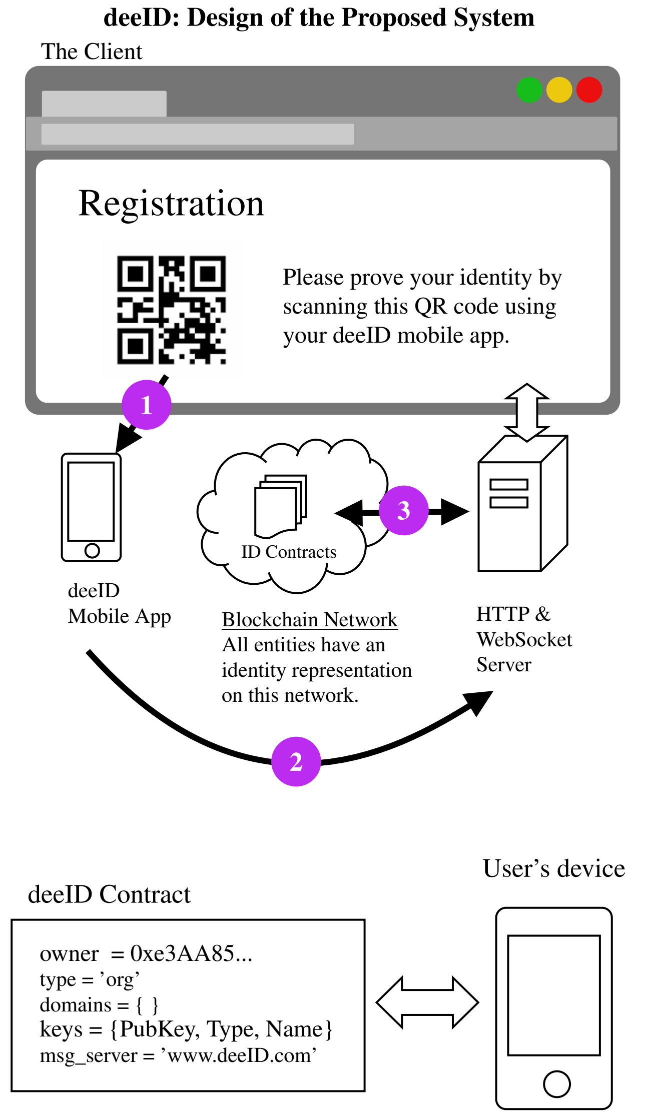

deeID: Design of the Proposed System
The Client
Registration
Please prove your identity by scanning this QR code using your deeID mobile app.
1
2
3
ID Contracts
deeID Mobile App
Blockchain Network
All entities have an identity representation on this network.
HTTP & WebSocket Server
deeID Contract
User's device
owner = 0xe3AA85...
type = 'org'
domains = { }
keys = {PubKey, Type, Name}
msg_server = 'www.deeID.com'

Figure 1: Overview of the deeID system. Ownership of the deeID contract (identity contract) can be proven via private-keys stored on the user's mobile phone device. Numeric labels are as follows. (1) Scan the QR code to get the necessary information about the server. (2) Interactive proof of identity (an identity that has been verified by an organisation, let that organisation be α). (3) Consult the blockchain for the existence of the identity issuer α; verify this organisation and return the decision to the client and mobile phone application.

A mobile phone-based out-of-band authentication scheme is resilient against Man-in-the-Browser and similar types of attacks and it can also provide superior security (compared to passwords) via a combination of *what we have* and *what we know* (e.g. protecting the mobile app with a pin).

The key components of the proposed system are as follows.

- *Blockchain and smart-contracts*. Blockchain is used as a decentralised public-key infrastructure. Each entity would have a unique representation on the network via a smart-contract and they can store public-keys associated with the user on this smart-contract.

- *Mobile phone application.* A device such as a mobile phone that can store private-keys and interact with services and servers over the internet (partly to ensure compatibility with existing web applications). The application uses QR codes to interact with web applications and then communicate with servers (using WebSocket connection) to carry out the necessary functions (e.g. interactive Fiat–Shamir scheme).

- *Libraries.* Required libraries that developers can install so that their users can use deeID to authenticate themselves on the developer's application.

We now use a normal identity card that we are all familiar with as an analogy to describe deeID and its functionalities. Figure 2 visualises this comparison; there are three main components to an identity card:

- *Photograph for verification.* We typically use the photograph of the individual on the identity card to identify the user. Meaning that the assumed valid ID card belongs to the individual whom is trying to prove their identity to the challenger. In deeID this is done through the ownership of a private-key associated with an identity. Via the Fiat–Shamir scheme one would be able to prove their ownership of the private-key and therefore identity. Ownership of this key means that some entity vouches for this person's claim of identity.

- *Human and machine readable strings.* We require a unique string that would uniquely identify the card. This string can also be used for record keeping with respect to the identity holder. Along with this unique string we have the full name of the user as well. Though it is typically not unique. However, it is what we use to identify with one another at a human level. In deeID, the smart-contract on the network with its unique address represents the unique machine-readable identity.

- *Identity issuer's integrity.* The integrity of the identity card comes through trusting the physical card itself and its physical properties. This trust also assumes that the verifier trusts the organisation that has issued the card. We find organisations that require different proofs of identity, e.g. passport and driving license in the UK. Therefore, we are concerned with different issuers and their reputation. Reputation in itself would require a full research paper, in the current state of deeID we have used a simple link on the identity issuer's smart-contract, which links to their web address if they have one proving that the owner has access to both.

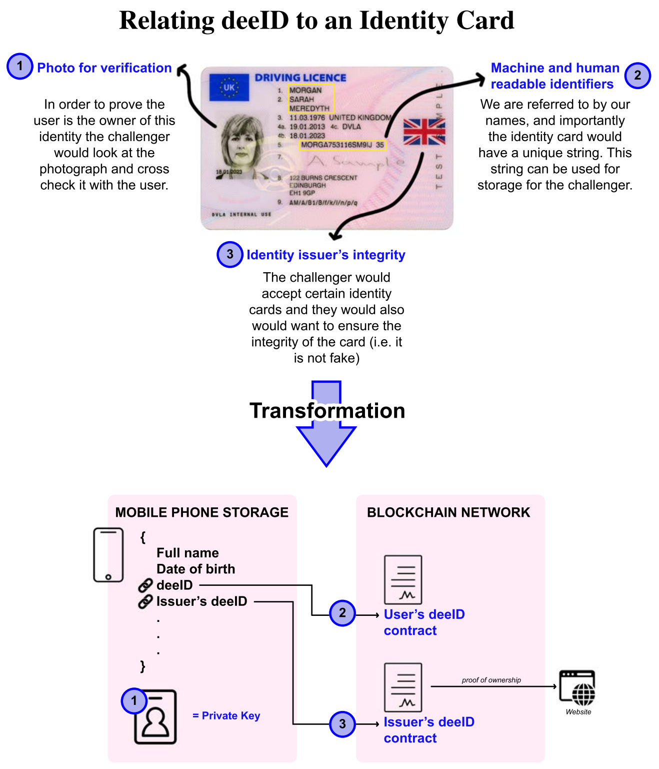

Relating deeID to an Identity Card
1 Photo for verification
In order to prove the user is the owner of this identity the challenger would look at the photograph and cross check it with the user.
DRIVING LICENCE
UK
1. MORGAN
2. SARAH
MEREDYTH
3. 11.03.1976 UNITED KINGDOM
4a. 19.01.2013 4c. DVLA
4b. 18.01.2023
5. MORGA753116SM9IJ 35
8. 122 BURNS CRESCENT EDINBURGH EH1 9GP
9. AM/A/B1/B/f/k/l/n/p/q
DVLA INTERNAL USE
TEST SAMPLE
Machine and human readable identifiers 2
We are referred to by our names, and importantly the identity card would have a unique string. This string can be used for storage for the challenger.
3 Identity issuer's integrity
The challenger would accept certain identity cards and they would also would want to ensure the integrity of the card (i.e. it is not fake)
Transformation
MOBILE PHONE STORAGE
{
Full name
Date of birth
deeID
Issuer's deeID
}
1
= Private Key
BLOCKCHAIN NETWORK
2
User's deeID contract
3
Issuer's deeID contract
proof of ownership
Website

Figure 2: At the top we have the main components of an identity card and at the bottom we have the equivalent functions in deeID.

3.1 Public-key Management

Management of public-keys is an important aspect of deeID which defines much of its capabilities. One novel contribution of this paper is the extension of the Fiat–Shamir scheme to multiple identity issuers. We achieve this by using a blockchain network as a decentralised PKI. Numerous studies have contributed to decentralised PKI (DPKI) [33] [34] [35] [36] and here we represent a simple solution built on Ethereum for deeID.

The use of DPKI goes beyond the Fiat–Shamir scheme in deeID. The user can use it to associate other public-keys to their identity, e.g. RSA keys for encryption and public-keys used on other devices that links back to the identity. Generally, DPKI has been proposed to be used for alternative DNS solutions [37], IoT [38] and much more. The challenges to the adoption of blockchain as a PKI is similar to that of any other blockchain application: consensus schemes, practicality, scaling and standardisation.

We assume that the user has ownership control of their deeID smart-contract and thus any changes within the contract are not malicious. Furthermore, the integrity of the the public-keys fall under the actual deeID and individual public-keys are not issued certificates as such. So, if a verifier has no knowledge of the user's deeID then they must first identify the deeID user (through the Fiat–Shamir scheme).

Adding new public-keys to the deeID smart-contract is a simple process of invoking the `addKey()` function with the following arguments: `title`, `key` and `comment`. The criteria for adding the key is that the user must be the owner of the deeID, i.e. by cross-checking the sender's address with the address of the owner (stored on the smart-contract). A real world-application could use any of the keys stored on the deeID to verify the integrity of the sender. Lastly, verifying a specific key would involve querying the blockchain with respect to the public-key and the deeID; this is achieved by having a separate function on the identity smart-contracts, `getKey()`, which would return all the details related to that key.

Other blockchain-based PKIs are more traditional in that they use certifications, e.g. CertLedger [33] and CertCoin [39]. On deeID the authenticity and security comes from the identification of the deeID (identity smart-contract) itself which can depend on the identity issuer and the verifier's willingness to trust the issuer or not. We do not propose a design to quantify this trust here, we envision the existence of a quantifiable reputation layer.

3.2 Authentication

The proposed system provides an alternative authentication option to password-based schemes. This alternative scheme is compatible with current technologies with minimum development overhead. The components of the authentication system are: a mobile phone running deeID application and the web application running a blockchain node with the necessary deeID libraries for it to interact with the mobile phone.

One can authenticate themselves via four different methods:

1. *Explicit link to deeID.* With this method we are using the identity smart-contract to authenticate the user. Essentially using the PKI method. Once the user has a public-key stored on their deeID contract then they can use that to authenticate themselves using their deeID address (smart-contract address). The user and the verifier have to query the chain for the existence of the specific public-key used to sign the authentication message. Lastly, because only the deeID address is used to authenticate the user, the user can store multiple public-keys on the contract and use multiple-devices with different keys for authentication. The Figure in Appendix E provides a visual sequence of main events that are required in this authentication process. At the end the typical session creation occurs.

2. *Generate a public-key.* One can also avoid the use of the blockchain all-together and simply generate a unique public-private key once they register with a website and use this public-key for authentication. The benefit of this is that the user can remain anonymous to the application. This also provides *unlinkable authentication*. This means that two or more colluding servers are unable to determine if the user is using their platform. Therefore, providing some privacy if required. Though the colluding servers could use other means, such as IP address, to guess if they have the same users.

3. *Fiat–Shamir identification.* The user can simply use the Fiat–Shamir scheme to authenticate themselves too. With this method you're also sharing your identity and its proof; this is therefore not useful if the user wishes to remain anonymous.

4. *Strong password.* Just like with the second method, one can use their mobile phone like a password manager and generate strong passwords to authenticate themselves with a given service.

Auth Method	Chain Storage	Query Chain	Interactive	Multi Device
Explicit link to deeID	Yes	Yes	No	Yes
Generate a public-key	No	No	No	No
Fiat–Shamir iden'	Yes	Yes	Yes	No
Strong password	No	No	No	No

Table 1: Methods of authentication with deeID.

Table 1 summarises the above authentication methods. Although the choice of which authentication method to use is not entirely dependent on the user, we believe, ultimately, online behaviour will dictate which method is best suited to a given application. The differences between the authentication methods are primarily the use of blockchain and ease of use (e.g. multi-device and requirement of a mobile phone).

3.3 Issuing an Identity

We have described the system as dynamic and self-serving. This means that anyone and anything can issue an identity. This is done through the Fiat–Shamir cryptosystem. Essentially, each issuer would have two large primes numbers that are kept secret, the product of these two are kept on the issuer's blockchain identity, i.e. smart-contract, so, when an identity holder claims to have been issued an identity, the verifier can check the existence of the product of the two large prime numbers within the issuers smart-contract or deeID$_{contract}$.

Below we go through an example of how an identity is issued by any other participant on the network. The first box states the requirements and the necessary steps required before two participants can create identities for each other.

Everyone in the system would have access to a function (described as f throughout the paper) that any developer can use to transform an identity string to a Fiat–Shamir public-key given the identity string I, public key indices and n. A simple numerical example is given in Appendix B.

When an identity is issued, the state of the blockchain would not change; therefore there is no blockchain related costs associated with issuing new identities. The issuer would have the product of p and q, n, within their smart-contract already, if not then the issuer would have to create a transaction and place n within their `keys` array on the smart-contract, in this scenario there is an associated cost with this transaction.

The following boxes will explain the steps required in issuing an identity. The end result is that the user will have a private-key associated with the verifier; the user can use this private-key to prove the identity that the issuer has vouched for.

Requirements

Steps required before an entity can issue an identity: Person **A**: The individual wishing to get a new identity from another person or organisation. Person **B**: Identity issuer.

1. **A**: Must create a deeID representation on the blockchain, e.g. on Ethereum, create a unique contract as per the common standards.
2. **B**: Must also create a deeID representation on the network.
3. **B**: Create a random modulus which is a product of two large prime numbers, $n = pq$, and n should be at least 512bits.
4. **B**: Store n on the network. In our implementation it is stored in a data structure on the deeID contract.

Example of issuing an identity

1. **A:** Hand over the required identity bits, $I_1 = \{\text{Fullname, DoB, deeID, City of Birth}\}$
2. **B:** Add a random string to I_1, and let that be I hereafter.
3. **B:** Verify the identity of A (e.g. face to face by looking at passport etc.)
4. **B:** If successful, create the credentials as follows:
   (a) Create the public-keys: $v_j = f(I, j)$ where v_j is the public-key, f is some pseudo-random function *(expansion of this is given in Appendix A and construction of such a function is explained in [40])* and j are small random values; to create the public keys we pick k distinct j values for which v_j is a quadratic residue mod n, i.e. solution exists for $x^2 = v_j \mod (n)$
   (b) Calculate the secret keys for A by taking the smallest $s_j = \sqrt{(v_j^{-1})} \mod (n)$.
   (c) So the set of k public keys and associated j values would give us two sets of values: $V = \{v_{j(1)}, v_{j(2)}, ..., v_{j(k)}\}$ which is our public keys, $J = \{j_1, j_2, ..., j_k\}$
5. Now B can hand over $\{I,J,S,V\}$ to A. V is technically not required because they can recalculate it themselves if they have I and J.

3.4 Verifying an Identity

We have covered how you can issue an identity to someone, now we will go over how you can verify that identity. You can think of verifying an identity via deeID as checking the person's passport to get the real identity of that person. So, To give context, this would typically happen when one is applying for a credit card, insurance policy or car finance online. Figure 3 provides an overview of how we will go about this. We assume that the verifier (the backend and WebSocket servers) will have a set of trusted identity issuers' deeID addresses (i.e. a list of entities that the website trusts in issuing out identity credentials to users). Therefore, when the website requests a proof of identity they will ensure that the provided identity is provided by one of the trusted issuers. The actual identification process occurs between the web server and the user's mobile phone device. For this to happen the user has to first interact with the client, capture the required information (via QR code) and then carry out the identification protocol with the WebSocket server. Upon the completion of the verifi-

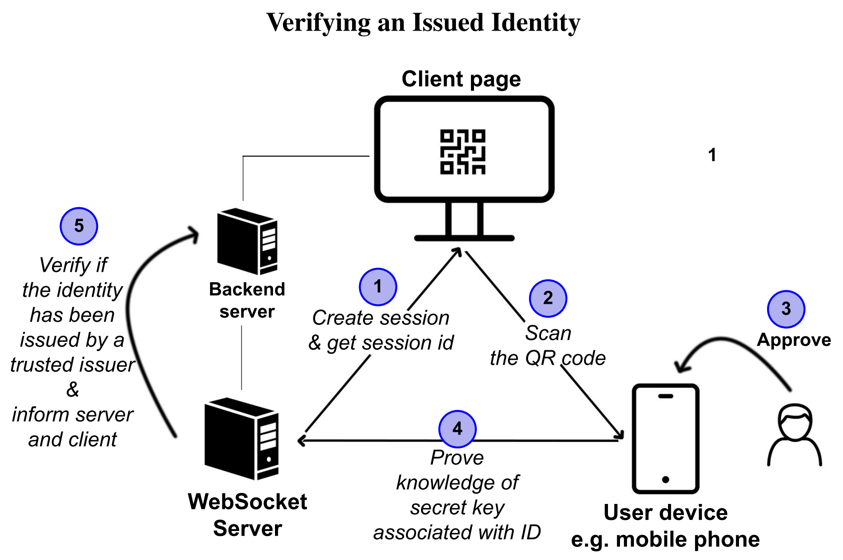

Verifying an Issued Identity
Client page
1
Backend server
WebSocket Server
User device e.g. mobile phone
1 Create session & get session id
2 Scan the QR code
3 Approve
4 Prove knowledge of secret key associated with ID
5 Verify if the identity has been issued by a trusted issuer & inform server and client

Figure 3: Visualisation of how a web page can verify a user's identity.

cation the server can update the client and carry out with the intended functions (e.g. registration and creating a session for the user).

3.5 Phishing Protection

This is a common threat that involves social engineering techniques to hijack identity and sensitive information, such as passwords. The threat channels are commonly: SMS or any instant messaging platform, emails, miss-typed or look-alike URLs. The consequences of phishing attacks are: account hijacking, identity theft, financial loss and much more. There are countermeasure techniques [41], however the lack of adoption of a single standard hinders the growth of these techniques.

deeID can circumvent phishing attacks, it does so primarily by mitigating the risks of social engineering; deeID reduces operator errors. Upon registration the deeID mobile application creates a link in its local storage. This store is essentially a dictionary of services that the user has a connection with. Because all communications and transfer of information is through the deeID mobile application, the threat of sharing sensitive information is drastically reduced. We assume in this scenario that the user's device is secure. The system avoids relying on the user for the entire process of authentication (recognising the website, inserting information and so on), this is similar to the *PhoolProof* system [42].

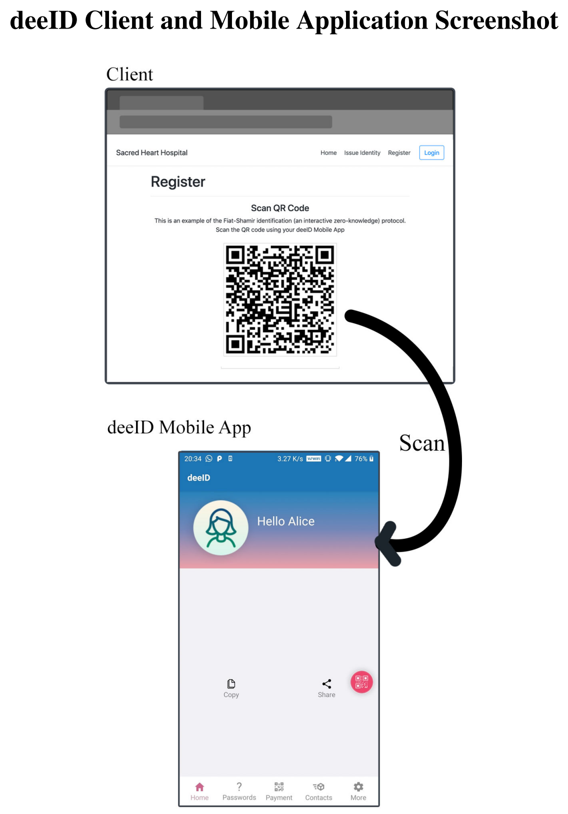
4 Implementation

We implemented a proof-of-concept that demonstrates the authentication and identification functionalities on the web. The mobile phone application was developed using the `react-native` [43] framework. The blockchain and smart-contracts were developed and tested on a local Ethereum [44] network. The Fiat–Shamir cryptosystem was implemented using `Python`. Dummy website representing an organisation (to issue identities and perform test authentication and identification) was created using the `flask` [45] web framework. The interaction between the back-end server of the website, client page and the mobile phone was carried out by a secondary back-end WebSocket server.

We implement the functionalities to allow one to authenticate themselves via public-key cryptography with a web server and to identify themselves via the Fiat–Shamir cryptosystem. Figure 4 shows the interfaces of the registration page (client) and the mobile phone application. The content of the QR code is show in Listing 1.

```
{
    'type': 'deeIDFS', //Fiat--Shamir identification
    'wsURL': 'https://localhost:5678', // WebSocket URL
    'uID': '', // session ID
    'expirytime' : '',
    'webdeeID': '', // deeID of the website
    'sig': '', // sig to ensure msg integrity
}
```

Listing 1: The above `JSON` is encoded into a QR code. This is used on the registration page for identification.



Sacred Heart Hospital

Register

Scan QR Code

This is an example of the Fiat-Shamir identification (an interactive zero-knowledge) protocol.
Scan the QR code using your deeID Mobile App

Figure 4: Client and mobile application interface design. Upon scanning the QR image the mobile application would recognise the message type and respond appropriately. In this instance it is an identification request.

4.1 Fiat–Shamir Implementation

The user can create their identities and receive Fiat–Shamir keys via the created website and then import it into their mobile phone application. The keys are 512bits long. The prover's functionalities are found on the mobile phone application and the verifier's functionalities on the WebSocket server.

4.2 Blockchain Functionalities

Our implementation begun with building the necessary blockchain components (such as smart-contracts). `Truffle` [46] was used to develop and test out the necessary identity contracts on a local Ethereum test network. In the deeID ecosystem each entity has a unique representation on the network via a smart-contract. The address of this smart-contract is the user's unique identifier. The main functionalities of this contract are shown in Listing 2.

```
    // Add a key to your contract
    function addKey(string memory _title,
    string memory _key, string memory _comment) {}

    // returns the length of keys array
    // i.e. number of keys stored
    function lenKeys() {}

    // Get a specific key
    // Returns title, key, status, comment and approver
    function getKey(uint _index) {}

    // Get the URL of the user's messaging server
    function msgServer() {}

    // change the messaging server
    function changeMsgServer(string memory _url) {}
```

Listing 2: Skeleton of the main deeID contract. Showing main functions, each user would have a unique deeID contract that other users can interact with.

4.3 Mobile Application Functionalities

The mobile phone application can carry out the interactive proof (as per the Fiat–Shamir cryptosystem) with a WebSocket server. The application can sign messages (for authentication) or other purposes with public-keys associated with their identity (or it can be anonymous too). Example of the payload used is shown in Listing 3.

```
var payload = JSON.stringify({
    'type': 'loginSig', // type of msg
    'uID': uID, // session ID
    'deeID': deeID, // user's deeID
    'expirytime': '',
    'data': '',
    'sig' : ''
});
```

Listing 3: Data structure used to communicate with the WebSocket server when the mobile phone application is attempting authentication.

4.4 WebSocket Functionalities

The WebSocket creates a dynamic and user friendly application as it allows the interaction between the server, client and mobile phone application to be live. The WebSocket server accepts `JSON` messages that have a `type` field. Currently it differentiates between a basic signature (created using the Ethereum crypto functions) and the Fiat–Shamir messages.

5 Evaluation

In order to evaluate our system we have used the framework proposed in [8], this allows us to compare the system with different types of authentication schemes in detail.

5.1 Semi-structured Evaluation of User Authentication Schemes Framework

In *'The Quest to Replace Passwords'* [8], the authors suggest a framework that compares various authentication schemes, with a strong focus on web authentication. In this section we will put deeID to the test.

Starting with *Usability benefits*, we believe deeID provides *Memorywise-Effortless* benefits, there is no need to remember a password or a secret. However, for more security we encourage users to have a pin-protected device. The application is highly scalable for the users, they can store as many credentials as they want without any extra burden, therefore better than passwords. However, the users are required to carry a device with themselves, such as a mobile phone device, since this is an object that they carry around with themselves anyway, we rate it as *Quasi-Nothing-to-Carry*. The scheme is also *Physically-Effortless*, they are only required to aim their device for example and then press a button to grant or decline a request. The scheme is also *Easy-to-learn* and intuitive, it requires scanning an image and pressing a button. Also it is far more efficient than a password, *Efficient-to-Use*, the user is only required to scan and press a button, no need to type or remember a password for example. *Infrequent-Errors*, mainly compared against biometric schemes, the process is very reliable and there are no false-positives. The scheme, however, is not so easy with respect to recovery of lost private keys, depending on which keys are lost, usually there are no recovery methods but to replace the lost credentials with the identity provider.

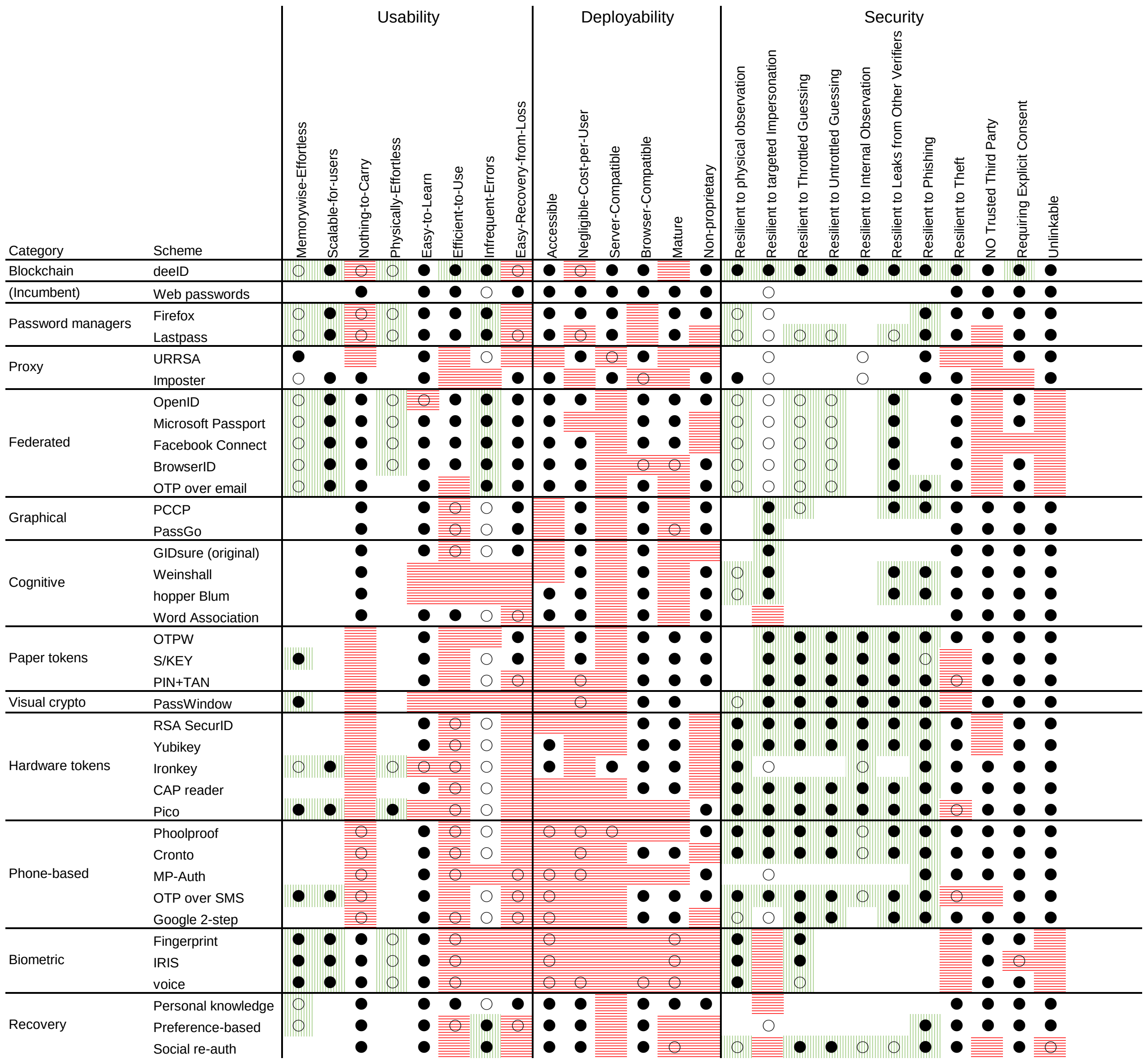
Moving onto *Deployability benefits*: The scheme is highly *Accessible*, and provides *Negligible-Cost-per-User*, beyond the access of a mobile-phone device the user is not required to spend anymore money, this is same for the verifier. The scheme is also *Server-Compatible*, meaning with their existing technology they are able to run the scheme. At the prover's end, they do not require to change or amend their browsers, therefore it is *Browser-Compatible*. It is not proprietary technology, however, it isn't mature at this stage.

On *Security-benefits*: It is resilient to physical observations, as the secret is not observable and all authentications do not require the user to reveal the secret keys. It is also resilient to targeted impersonation, knowledge of ones personal details will not mean that they can break the authentication scheme. Given that the scheme is challenge-response based, and it is machine-to-machine (mobile phone to server), the rate of failure due to any other reason than wrong key is low, therefore the chance of guessing and getting it right is very low. Unthrottled-guessing depends on the identity provider and the length of their keys. Resilient-to-Internal-Observation is very low, since the user does not interact with the private keys directly, therefore key logging and other methods have no chance of revealing the secret-key. Given the zero-knowledge concept of the Fiat–Shamir concept, anything that the verifier knows cannot help reveal any information about the user's secret key.The scheme is also resilient to phishing attacks, again due to ZK, no information about the secret-key is revealed. The scheme is less resilient to theft, however with the use of a PIN on the device, the scheme is qusi-resilient-to-theft. *No-Trusted-Third-Party*: the scheme does not rely on trusted third-parties beyond the initial identity offer, therefore they have no control and influence thereafter, if they become compromised it only means that identities should no longer be accepted by the newly untrusted-third-party. The scheme requires Explicit consent from the user for any authentication (physically pressing a button after logging in to their device). Since there is no authenticator, they cannot link whether the same user is logging in to various services. However, if the verifiers do collude and share user information, then of course they will be able to know that they have the same set of users. The point being that with just the authentication scheme itself, they will not be able to link users.

6 Discussion

Human identification is an integral part of the security of the digital ecosystem. Though it is a missing element at the moment which is leading to account breaches, identity theft and much more. In this paper we introduced the Fiat–Shamir scheme and have generalised it so that it could work on a larger scale (country and global level) with *multiple* identity issuers. Our solution has been to use the blockchain as a decentralised PKI. Blockchain-based PKIs have shown promises in research and applications. Emergence of blockchain-based PKIs have come about mainly due to the weak-link security problem with respect to certificate authorities and the X.509 standard [47] (other main PKI standard is OpenPGP [48]). Consequently, this has led to new approaches such as the Google Certificate Transparency project [49]. Blockchain PKI models such as [50] attempt to overcome the weaknesses of the current PKI standards.

We begun our quest with the question of how to represent a human on a digital network? and what mechanisms are required to interact with the identity of this entity? Going back to our four rules: a) Capture: each identity must have a unique representation on the network. Each deeID contract has a unique address. b) Recording: It must be machine and human readable. The deeID contract address is unique and machine-readable. The user can also prove their identity and give their human readable names to the challenger. c) Security: can we protect from impersonation and other attacks?



		Usability								Deployability						Security										
Category	Scheme	Memorywise-Effortless	Scalable-for-users	Nothing-to-Carry	Physically-Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-proprietary	Resilient to physical observation	Resilient to targeted Impersonation	Resilient to Throttled Guessing	Resilient to Untrottled Guessing	Resilient to Internal Observation	Resilient to Leaks from Other Verifiers	Resilient to Phishing	Resilient to Theft	NO Trusted Third Party	Requiring Explicit Consent	Unlinkable
Blockchain	deeID	○ 🟩	● 🟩	○ 🟥	○ 🟩	●	● 🟩	● 🟩	○ 🟥	●	○ 🟥	●	●	🟥	●	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	●	● 🟩	●
(Incumbent)	Web passwords			●		●	●	○	●	●	●	●	●	●	●		○						●	●	●	●
Password managers	Firefox	○ 🟩	● 🟩	○ 🟥	○ 🟩	●	●	● 🟩	🟥	●	●	●	🟥	●	●	○ 🟩	○					● 🟩	●	●	●	●
	Lastpass	○ 🟩	● 🟩	○ 🟥	○ 🟩	●	●	● 🟩	○ 🟥	●	○ 🟥	●	🟥	●	🟥	○ 🟩	○	○ 🟩	○ 🟩		○ 🟩	● 🟩	●	🟥	●	●
Proxy	URRSA	●		🟥		●	🟥	○	🟥	🟥	●	○ 🟥	●	🟥	🟥		○			○		●	🟥	🟥	●	●
	Imposter	○	●	●		●	🟥	🟥	●	●	🟥	●	○ 🟥	🟥	●	●	○			○		●	●	🟥	🟥	●
Federated	OpenID	○ 🟩	● 🟩	●	○ 🟩	○ 🟥	●	● 🟩	●	●	●	🟥	●	●	●	○ 🟩	○	○ 🟩	○ 🟩		● 🟩		●	🟥	●	🟥
	Microsoft Passport	○ 🟩	● 🟩	●	○ 🟩	●	●	● 🟩	●	●	🟥	🟥	●	●	🟥	○ 🟩	○	○ 🟩	○ 🟩		● 🟩		●	🟥	●	🟥
	Facebook Connect	○ 🟩	● 🟩	●	○ 🟩	●	●	● 🟩	●	●	●	🟥	●	●	🟥	○ 🟩	○	○ 🟩	○ 🟩		● 🟩		●	🟥	🟥	🟥
	BrowserID	○ 🟩	● 🟩	●	○ 🟩	●	●	● 🟩	●	●	●	🟥	○ 🟥	○ 🟥	●	○ 🟩	○	○ 🟩	○ 🟩		● 🟩		●	🟥	●	🟥
	OTP over email	○ 🟩	● 🟩	●		●	🟥	● 🟩	●	●	●	🟥	●	🟥	●	○ 🟩	○	○ 🟩	○ 🟩		● 🟩	● 🟩	●	🟥	●	🟥
Graphical	PCCP			●		●	○ 🟥	○	●	🟥	●	🟥	●	🟥	●		● 🟩	○ 🟩			● 🟩	● 🟩	●	●	●	●
	PassGo			●		●	○ 🟥	○	●	🟥	●	🟥	●	○ 🟥	●		● 🟩						●	●	●	●
Cognitive	GIDsure (original)			●		●	○ 🟥	○	●	🟥	●	🟥	●	🟥	🟥		● 🟩						●	●	●	●
	Weinshall			●		🟥	🟥	🟥	🟥	🟥	●	🟥	●	🟥	●	○ 🟩	● 🟩				● 🟩	● 🟩	●	●	●	●
	hopper Blum			●		🟥	🟥	🟥	🟥	●	●	🟥	●	🟥	●	○ 🟩	● 🟩				● 🟩	● 🟩	●	●	●	●
	Word Association			●		●	●	○	○ 🟥	●	●	🟥	●	🟥	●		🟥						●	●	●	●
Paper tokens	OTPW			🟥		●	🟥	🟥	●	🟥	●	🟥	●	●	●		● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	●	●	●	●
	S/KEY	● 🟩		🟥		●	🟥	○	●	🟥	●	🟥	●	●	●		● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	○ 🟩	🟥	●	●	●
	PIN+TAN			🟥		●	🟥	○	○ 🟥	🟥	○ 🟥	🟥	●	●	●		● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	○ 🟥	●	●	●
Visual crypto	PassWindow	● 🟩		🟥		🟥	🟥	🟥	🟥	🟥	○ 🟥	🟥	●	●		○ 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	🟥	●	●	●
Hardware tokens	RSA SecurID			🟥		●	○ 🟥	○	🟥	🟥	🟥	🟥	●	●	🟥	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	●	🟥	●	●
	Yubikey			🟥		●	○ 🟥	○	🟥	●	🟥	🟥	●	●	🟥	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	●	🟥	●	●
	Ironkey	○ 🟩	● 🟩	🟥	○ 🟩	○ 🟥	○ 🟥	○	🟥	●	🟥	●	●	●	🟥	● 🟩	○			○ 🟩		● 🟩	●	●	●	●
	CAP reader			🟥		●	○ 🟥	○	🟥	🟥	🟥	🟥	●	●	🟥	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	●	●	●	●
	Pico	● 🟩	● 🟩	🟥	● 🟩	🟥	○ 🟥	○	🟥	🟥	🟥	🟥	🟥	🟥	●	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	● 🟩	○ 🟥	●	●	●
Phone-based	Phoolproof			○ 🟥		●	○ 🟥	○	🟥	○ 🟥	○ 🟥	○ 🟥	🟥	🟥	●	● 🟩	● 🟩	● 🟩	● 🟩	○ 🟩	● 🟩	● 🟩	●	●	●	●
	Cronto			○ 🟥		●	○ 🟥	○	🟥	🟥	○ 🟥	🟥	●	●	🟥	● 🟩	● 🟩	● 🟩	● 🟩	○ 🟩	● 🟩	● 🟩	●	●	●	●
	MP-Auth			○ 🟥		●	○ 🟥	🟥	○ 🟥	○ 🟥	○ 🟥	🟥	🟥	🟥	●		○					● 🟩	●	●	●	●
	OTP over SMS	● 🟩	● 🟩	○ 🟥		●	🟥	○	○ 🟥	○ 🟥	🟥	🟥	●	●	●	● 🟩	● 🟩	● 🟩	● 🟩	○ 🟩	● 🟩	● 🟩	○ 🟥	🟥	●	●
	Google 2-step			○ 🟥		●	○ 🟥	○	○ 🟥	○ 🟥	🟥	🟥	●	●	🟥	○ 🟩	○	● 🟩	● 🟩		● 🟩	● 🟩	●	●	●	●
Biometric	Fingerprint	● 🟩	● 🟩	●	○ 🟩	●	○ 🟥	🟥	🟥	○ 🟥	🟥	🟥	🟥	○ 🟥	🟥	● 🟩	🟥	● 🟩					🟥	●	●	🟥
	IRIS	● 🟩	● 🟩	●	○ 🟩	●	○ 🟥	🟥	🟥	○ 🟥	🟥	🟥	🟥	○ 🟥	🟥	● 🟩	🟥	● 🟩					🟥	●	○ 🟥	🟥
	voice	● 🟩	● 🟩	●	○ 🟩	●	○ 🟥	🟥	🟥	○ 🟥	○ 🟥	🟥	○ 🟥	○ 🟥	🟥	● 🟩	🟥	○ 🟩					🟥	●	●	🟥
Recovery	Personal knowledge	○ 🟩		●		●	●	○	●	●	●	🟥	●	●	●		🟥						●	●	●	●
	Preference-based	○ 🟩		●		●	○ 🟥	● 🟩	○ 🟥	●	●	🟥	●	🟥	🟥		○					● 🟩	●	●	●	●
	Social re-auth			●		●	🟥	● 🟩	🟥	●	●	🟥	●	○ 🟥	🟥	○ 🟩	🟥	● 🟩	● 🟩	○ 🟩	○ 🟩	● 🟩	●	🟥	●	○ 🟥

●= offers the benefit; ○= almost offers the benefit; *no circle* = does not offer the benefit.
🟩= better than passwords; 🟥= worse than passwords; *no background pattern* = no change.
We group related schemes into categories. For space reasons, in the present paper we describe at most one representative scheme per category; the companion technical report [1] discusses all schemes listed.

Figure 5: Visual evaluation of various schemes and our new blockchain-based scheme (deeID)

Best current method is using public-key cryptography and *what we posses* rather than *what we know* or *what we are*. d) Verification: Using the mobile phone device and the Fiat–Shamir scheme we can verify identities easily. Similar to uPort we have represented the identity by a smart-contract which is uniquely identified in the network. Therefore, each identity has an interactive code on the network (managed by the identity holder themselves). So, here, identity is no longer a piece of string such as your full name, but a totally unique digital representation on a network. The benefit of this is that it is a single-source of truth that everyone can use and agree upon. So, your bank, hospital and school can all use the same identity. Using the same identity and authentication scheme can provide data sharing which in itself has huge benefits. We believe merging the Fiat–Shamir scheme with the existing blockchain identity methods is a novel method for an identification system.

The benefit of using the Fiat–Shamir scheme over simply signing an identity string is the privacy and security it provides. Moreover it saves transactions and space on the network. However, it is still important to keep our 'full names', the basic string identity, because we humans still need to communicate with one another and be able to identify each other. Which is not possible if we are identified by a 512bit unique identity string. There have been studies such as [51], *'Towards reliable storage of 56-bit secrets in human memory'*, where the authors tried to see if we humans can remember long bits of string. The conclusion was that humans are able to learn cryptographic secrets (56-bit in this case), though it requires significant learning periods and other limitations such as recalling times.

Trusting the identity issuer depends upon the verifier. At the moment it is conceived to be a simple linear trace back of identities until the final link can be verified by an official web page. In our implementation we have a simple link to an official website or regulatory-body portal. However, there is considerable work to be done in improving this, e.g. one can quantify trust and provide recommendation to the verifier whether the issuer is trust-worthy or not. A reputation-based network can provide some metric for verifiers to work from. However, one must consider the ethical implications of quantifying reputation.

The proposed system is an out-of-band system. With this characteristics we can naturally guard against Man-in-the-Browser (MitB) [52] and form grabber threats. deeID simply reads a QR code from the web client and then transfers the relevant information to another server using a WebSocket connection. This separate channel circumvents MitB type threats. A notable example of such a threat is the British Airways (BA) hack; BA is reportedly being fined £183 million [53] for the 2018 breach of their site where thousands of customer's personal information were stolen via a form grabbing technique. The attackers compromised a script on the site [54] and were able to capture data as it was being entered on the page. The deeID architecture prevents authentication and identification information from being stolen via these attacks, one can extend deeID for an out-of-band form transactions.

Our evaluation based on the framework from [8] indicates that between the three categories of usability, deployability and security, our protocol betters federated systems in security, betters hardware tokens and phone-based systems in deployability. According to the evaluation in [8], these systems are the closest systems in bettering password. Our protocol is inherently a password manager too, as the device used to manage keys and do the necessary computation on behalf of the identity holder. It can also store and manage passwords. But, this is merely a backward-compatible feature, as we envision a future that does not use password-based schemes.

It should be noted that the framework in figure 1 is designed for the practicality of a human as it has factors such as 'nothing to carry', 'physical-effortlessness' and 'memory-wise-effortless'. Therefore, the protocols are very human-centric at the moment. Nevertheless, as we progress into the domains of AI and IoT we can improve upon these protocols and frameworks to cater to other non-human entities too. Also note that no other system can better passwords that are stored and retrieved easily in our brains but as the number of passwords to remember grows then we start running into problems and thus other schemes such as password managers' benefits become more apparent.

Beyond the primary use case of authentication and identification on the web, we deem the following to be useful applications of deeID. SIM swap attack is where an attacker uses social-engineering (operator error and weakness) to convince the user's mobile carrier to swap the SIM to the attacker's SIM. The consequence of this is that the attacker can hijack accounts that use two factor authentication (2FA). Notable hacks include the hijacking of celebrity Instagram accounts [55] and individuals losing millions of pounds of cryptocurrency as their exchange accounts were linked to their mobile phone number using 2FA. This is a growing threat that is leading to more organised crime [56]. Preventive techniques include using a PIN or better 2FA [57]. However, these are patchy and preventive measures; a standardised solution through the use of decentralised identity like that of deeID across mobile network carriers would be a better solution. Explicitly linking the user's deeID to their SIM and having a hardware linked crypto keys.

The mechanism of preventing identity theft is as follows. An attacker can steal an identity and order a credit card by only knowing the victims name, address and date of birth. Credit checks, for example, should be a push mechanism from the identity owner rather than a pull by knowing the victim's basic personal information. With deeID the process of opening a credit account can be as follows: The credit agencies (Experian [58], Equifax [59] and TransUnion [60]) would encourage their customers to link their data to their deeID. Once a company does a credit check through these credit agencies, the agency would notify the identity owner and request approval rather than automatically granting access.

The system limitations are as follows. In the proposed system, trusted parties are not totally eliminated; we require these trusted entities to physically verify the user before issuing them an identity. This process would require some sort of a standard and perhaps an explanation to identity verifiers to instill trust in their (identity verification) process.

Our implementation used Ethereum and its proof-of-work consensus scheme. This consensus scheme has limitations in energy usage, trust of the miners. We believe alternative consensus schemes such as Proof-of-Stake or more experimental schemes such as Proof-of-Reputation [61] are better suited to a network exclusively built for identity management.

Using a blockchain system on a mobile phone is also a challenge as we do not verify the integrity of the network on the mobile phone. In our implementation we relied on a trusted node to get information regarding the state of the blockchain. An implementation of a light node would be a method of overcoming this challenge. This could form a further branch of future work.

Revoking an identity from a user could happen in reality. Currently, with the Fiat–Shamir scheme, if an issuer issues x number of identities based on the public n (product of p and q) then revoking a single identity means revoking identities for all individuals using n. This is clearly a challenge and issuing and re-issuing identities if one identity need revoking is not efficient.

7 Related Work

There are numerous examples of blockchain-based identity systems, each utilising either Ethereum [44] or Bitcoin as their blockchain platform. Moreover, organisations are coming together under foundations such as the *Decentralized Identity Foundation* [62] to create a standardised ecosystem. Standards are also being developed at the same time, evident by Decentralized Identifiers (DID) [63].

Numerous papers such as [64] have implemented variations of identity solutions on the blockchain. Zhu and Badr [65] in their survey provide further information on blockchain-based identity systems. Many of the notable implementations are commercial or experimental projects, such as uPort [66] and Sovrin [67]. Taking these two as an example, they both try to do the same function but using very different technology and philosophy to tackle decentralisation, security and trust. uPort uses the Ethereum blockchain; so, its consensus, security and growth is bounded by the Ethereum chain. Sovrin has taken a different approach, it uses the Hypderledger [68] stack and it is a permissioned ledger. With respect to consensus, the integrity of the Sovrin network is maintained by the so called stewards. Therefore, one has to raise the question of whether if it really is self-sovereign at all? The transparency of a cheated system is irrelevant if the honest participants cannot do anything about it.

The closest related work, to our knowledge, is the work of Boontaetae et al. [13]. They have similarly utilised the blockchain (Ethereum in their implementation) as a PKI system and identities are issued by trusted sources known as Trusted Source Certificate Authorities (TSCA). the TSCAs are responsible for verifying the end-user's identity via some offline means and then signing their cryptographic keys using their own. Consequently, storing the relevant keys in a universal smart-contract (named RDI). The cryptographic keys are known to be the hot and cold keys, where the hot-key represents an identity and the cold-key is responsible for revoking the hot-key. Several issues begin to emerge at this point, e.g. signing keys en mass proves to be a fatal security problem. If a key gets compromised then the entirety of the signed-keys must be re-signed by a new key. Another issue is the representation of identity, it is fair to say that an identity is represented by a key. But *who* is that key? and what human readable attributes can we attach to that key?. This hasn't been discussed enough in the paper - legitimacy of an identity can be important in its own. However, most applications will require cross-referencing and at times some human readable attributes too.

Whilst most of the above works provide some sort of authentication (e.g. Login with uPort) they do not provide the identification standard that we provide through the Fiat–Shamir cryptosystem. deeID has the ability to provide multiple identities with minimum use of the blockchain network.

8 Conclusion

In this paper we have demonstrated a clear use for the Fiat–Shamir identification cryptosystem at country and global level with the ability of having multiple identity issuers. We have designed and implemented an out-of-band and blockchain-based authentication and identification system (deeID) that can be a secure and practical alternative to password-based schemes. We have argued that deeID provides better security than federated, specific mobile phone-based, and hardware-based systems which in turn provide better security than passwords. Deployability is where most schemes fall back on compared to passwords. However, this is something that cannot be beaten due to *knowing something* is more deployable than all other systems. Given that our system can inherently be a password manager too, it therefore strikes a better balance between the three major comparison factors of usability, deployability and security.

deeID can also be built on top of other blockchain technologies. Our novel contribution is improving upon identity representation through smart-contracts, allowing the blockchain as a cryptographic key management and a trusted-source of identities and providing a mechanism for people on the network to provide identities to one another.

Future work can be separated into four different pillars: (1) General algorithm optimisation of the Fiat–Shamir cryptosystem. (2) We believe that an identity system such as the one proposed in this paper could become an integral part of future blockchain systems that utilise Proof-of-Authority consensus schemes. A deeID based Proof-of-Authority that is Byzantine Fault Tolerant with some quantifiable method for reputation to manage the network. (3) Extend the design to support machine and legal entity identity (i.e. go beyond human identity). (4) Work on interoperability of the design and communication standards.

Availability

Our implementation and code can be found at `https://github.com/deeId`.

References

[1] Robert Morris and Ken Thompson. Password Security: A Case History. *Commun. ACM*, 22(11):594–597, November 1979.

[2] Dinei Florencio and Cormac Herley. A Large-scale Study of Web Password Habits. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 657–666, New York, NY, USA, 2007. ACM. Banff, Alberta, Canada.

[3] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and Xiaofeng Wang. The tangled web of password reuse. In *NDSS*, 2014.

[4] Greg Wolfond. A blockchain ecosystem for digital identity: Improving service delivery in canada's public and private sectors. *Technology Innovation Management Review*, 7:35–40, Oct 2017.

[5] The 18 biggest data breaches of the 21st century | CSO Online. https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html.

[6] Identity theft statistics report. https://www.experian.com/blogs/ask-experian/identity-theft-statistics.

[7] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, Santa Barbara, California, USA, January 1986. Springer-Verlag.

[8] J. Bonneau, C. Herley, P. C. v Oorschot, and F. Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *2012 IEEE Symposium on Security and Privacy*, pages 553–567, May 2012.

[9] Namecoin. https://www.namecoin.org.

[10] Blockstack. https://blockstack.org.

[11] Conner Fromknecht and Dragos Velicanu. CertCoin : A NameCoin Based Decentralized Authentication System 6 . 857 Class Project. 2014.

[12] Guilherme Vieira Pinto, João Pedro Dias, and Hugo Sereno Ferreira. Blockchain-based PKI for crowdsourced IoT sensor information. In Ana Maria Madureira, Ajith Abraham, Niketa Gandhi, Catarina Silva, and Mário Antunes, editors, *Proceedings of the Tenth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2018)*, Advances in Intelligent Systems and Computing, pages 248–257. Springer International Publishing.

[13] P. Boontaetae, A. Sangpetch, and O. Sangpetch. RDI: Real digital identity based on decentralized PKI. In *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, pages 1–6, November 2018.

[14] Think smartphones are ubiquitous now? Just wait a few years | Charles Arthur | Technology | The Guardian. https://www.theguardian.com/technology/2012/jun/27/smartphones-iphone-mobile-market.

[15] Ubiquitous mobile phones are becoming indispensable | ACM Inroads. https://dl.acm.org/doi/10.1145/1963533.1963545.

[16] Combating synthetic identity fraud | McKinsey. https://www.mckinsey.com/business-functions/risk/our-insights/fighting-back-against-synthetic-identity-fraud.

[17] OpenID foundation website. https://openid.net.

[18] OAuth community site. https://oauth.net.

[19] Security assertion markup language. https://en.wikipedia.org/w/index.php?title=Security_Assertion_Markup_Language&oldid=905598209. Page Version ID: 905598209.

[20] Using OAuth 2.0 to access google APIs | google identity platform. https://developers.google.com/identity/protocols/OAuth2.

[21] Facebook. https://www.facebook.com.

[22] Mohammad Mannan and Paul C. van Oorschot. Leveraging personal devices for stronger password authentication from untrusted computers. *Journal of Computer Security*, 19:703–750, 2011.

[23] IronKey. https://www.ironkey.com/en-US.

[24] L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2019–2020, December 2003.

[25] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology*, Lecture Notes in Computer Science, pages 47–53. Springer, Berlin, Heidelberg, August 1984.

[26] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304. ACM.

[27] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, June 1988.

[28] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology — CRYPTO 2001*, Lecture Notes in Computer Science, pages 213–229. Springer, Berlin, Heidelberg, Aug 2001.

[29] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding*, Lecture Notes in Computer Science, pages 360–363. Springer, Berlin, Heidelberg, Dec 2001.

[30] L. C. Guillou and J.-J. Quisquater. A Practical Zero-knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In *Lecture Notes in Computer Science on Advances in Cryptology-EUROCRYPT'88*, pages 123–128, New York, NY, USA, 1988. Springer-Verlag New York, Inc. Davos, Switzerland.

[31] Claus P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '89, pages 688–689, Berlin, Heidelberg, 1990. Springer-Verlag. Houthalen, Belgium.

[32] A. J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. CRC Press, 1997.

[33] Murat Yasin Kubilay, Mehmet Sabir Kiraz, and Hacı Ali Mantar. CertLedger: A new PKI model with Certificate Transparency based on blockchain. *Computers & Security*, 85:333–352, August 2019.

[34] L. Dykcik, L. Chuat, P. Szalachowski, and A. Perrig. BlockPKI: An Automated, Resilient, and Transparent Public-Key Infrastructure. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 105–114, November 2018.

[35] A. Yakubov, W. Shbair, and R. State. BlockPGP: A Blockchain-Based Framework for PGP Key Servers. In *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 316–322, November 2018.

[36] Christos Patsonakis, Katerina Samari, Aggelos Kiayias, and Mema Roussopoulos. On the Practicality of Smart Contract PKI. *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pages 109–118, April 2019. arXiv: 1902.00878.

[37] E. Karaarslan and E. Adiguzel. Blockchain Based DNS and PKI Solutions. *IEEE Communications Standards Magazine*, 2(3):52–57, September 2018.

[38] A. Singla and E. Bertino. Blockchain-Based PKI Solutions for IoT. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 9–15, October 2018.

[39] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. A Decentralized Public Key Infrastructure with Identity Retention. *IACR Cryptology ePrint Archive*, 2014:803, 2014.

[40] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, August 1986.

[41] Ahmed Aleroud and Lina Zhou. Phishing environments, techniques, and countermeasures: A survey. *Computers & Security*, 68:160–196, July 2017.

[42] Bryan Parno, Cynthia Kuo, and Adrian Perrig. Phoolproof phishing prevention. In Giovanni Di Crescenzo and Avi Rubin, editors, *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 1–19. Springer Berlin Heidelberg, 2006.

[43] React Native · A framework for building native apps using React. https://facebook.github.io/react-native.

[44] Ethereum. https://ethereum.org.

[45] Flask. https://palletsprojects.com/p/flask.

[46] Truffle Suite. https://trufflesuite.com.

[47] X.509 - public key and attribute certificate frameworks. https://www.itu.int/rec/T-REC-X.509.

[48] David Shaw, Lutz Donnerhacke, Rodney Thayer, Hal Finney, and Jon Callas. Openpgp message format. https://tools.ietf.org/html/rfc4880.

[49] Certificate Transparency. http://www.certificate-transparency.org.

[50] Artem S. Konoplev, Alexey G. Busygin, and Dmitry P. Zegzhda. A Blockchain Decentralized Public Key Infrastructure Model. *Automatic Control and Computer Sciences*, 52(8):1017–1021, 2018.

[51] Joseph Bonneau and Stuart Schechter. Towards reliable storage of 56-bit secrets in human memory. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 607–623. USENIX Association, 2014. San Diego, CA.

[52] Najla Etaher, George R.S. Weir, and Mamoun Alazab. From ZeuS to Zitmo: Trends in Banking Malware. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 1386–1391, August 2015.

[53] British Airways faces record £183m fine for data breach. *BBC News*, July 2019.

[54] Jordan Bishop. This Is How 380,000 British Airways Passengers Got Hacked. https://www.forbes.com/sites/bishopjordan/2018/09/11/how-british-airways-got-hacked.

[55] Karissa Bell. Instagram users are reporting the same bizarre hack. https://mashable.com/article/instagram-hack-locked-out-of-account.

[56] Lorenzo Franceschi-Bicchierai. The SIM Hijackers. https://www.vice.com/en_us/article/vbqax3/hackers-sim-swapping-steal-phone-numbers-instagram-bitcoin, July 2018.

[57] How to Protect Your Phone Against a SIM Swap Attack. wired. https://www.wired.com/story/sim-swap-attack-defend-phone.

[58] Experian plc - Credit Agency. https://www.experianplc.com.

[59] Equifax UK - Credit Agency. https://www.equifax.co.uk.

[60] TransUnion - Credit Scores, Credit Reports & Credit Check. https://www.transunion.com.

[61] Qianwei Zhuang, Yuan Liu, Lisi Chen, and Zhengpeng Ai. Proof of Reputation: A Reputation-based Consensus Protocol for Blockchain Based Systems. In *Proceedings of the 2019 International Electronics Communication Conference*, IECC '19, pages 131–138, New York, NY, USA, 2019. ACM.

[62] DIF - Decentralized identity Foundation. https://identity.foundation.

[63] Decentralized Identifiers (DIDs) v1.0. https://www.w3.org/TR/did-core/#a-simple-example.

[64] Y. Liu, Z. Zhao, G. Guo, X. Wang, Z. Tan, and S. Wang. An identity management system based on blockchain. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 44–4409.

[65] Xiaoyang Zhu and Youakim Badr. Identity management systems for the internet of things: A survey towards blockchain solutions. *Sensors*, 18(12):4215, December 2018.

[66] uPort.me. https://www.uport.me.

[67] Sovrin. https://sovrin.org.

[68] Hyperledger – Open Source Blockchain Technologies. https://www.hyperledger.org.

[69] Daniel Shanks. Five number-theoretic algorithms. *Proceedings of the Manitoba Conference on Numerical Mathematics and Computing.*, VII:51–70, 1973.

[70] Jan-Christoph Schlage-Puchta. On Shanks' Algorithm for Modular Square Roots. *arXiv:1105.1456*, May 2011.

Appendices

A Pseudo-random functions and the j values

In explaining the identification cryptosystem, Fiat and Shamir, in their paper [7] use the so called pseudo-random function along with a random seed, j. The pseudo-random function f maps an arbitrary string to the range of $[0,n)$, where n is the product of the prime numbers p and q. A pseudorandom function (PRF), essentially, emulates a random oracle in that the output of both should be *indistinguishable* by a polynomially bounded computation. Therefore, all outputs of the pseudorandom function will appear to be random but it of course is a deterministic function.

The j value which is used in calculating the public key, $v_j = f(I, j)$ (I is our identity string), j is the random seed in our input to the pseudorandom function. Moreover it acts as an adjustment factor to get a public-key (v_j) that is a quadratic residue $\mod n$.

B From Public-key to Private-key Using the Fiat–Shamir Scheme

Here we will go through a numerical example of how to calculate the private-key key from the public-key as per the Fiat–Shamir scheme [7].

Remarks: The actual cryptosystem uses a function that maps the identity string to a range of $[0,n)$. We will skip this step. Lastly, the cryptosystem uses a set of k keys, we will create one key in this example (i.e. $k = 1$).

We start with our secret prime numbers of p and q:

$$p = 7$$

$$q = 11$$

$$n = p * q = 77$$

lets assume that $v = f(I,k) = 64$, thus our public key is 64, which is a quadratic residue $\mod n$. Now our goal is to compute the smallest s:

$$s = \sqrt{v^{-1}} \mod n$$

Calculate $\bar{v} = v^{-1} \mod n$ using the extended euclidean algorithm:

$$\bar{v} = 64^{-1} \mod 77$$

$$\bar{v} = 71$$

Now we find the smallest square root for this value, $\bar{v}$.

$$71 \mod 7 = 1$$

$$71 \mod 11 = 5$$

Now calculating the square roots via the Tonelli–Shanks algorithm we obtain ± 1 and ± 4. Next, we use the Chinese-remainder theorem to combine the roots to find the square roots $\bar{v} \mod n$. And thus for our values of ± 1 and ± 4 we obtain 15. Thus $s = 15$. The python function that we implemented to create the Fiat–Shamir private-key is shown in Appendix C.

C Python Function: Generating Fiat–Shamir Secret-keys from Public-keys

In Appendix B we went through a simple mathematical example of creating a private-key from the public-key based on the Fiat–Shamir cryptosystem. In Listing 4 we show the Python code to do the same computation.

```
'''
 GENERATE SECRET KEY
     v = public key
     p and q = secret factors of n
     n = modulus, product of p & q
 '''
def genSecretKey(v, p, q, n):
    v = egcd(v, n)[1] % n
    b1 = tonelli(v % p, p)
    b2 = tonelli(v % q, q)

    # Square root signs
    a = [b1, b2, b1, b2*-1, b1*-1,
        b2*-1, b1*-1, b2]
    j = 0
    smallest = -1
    for i in range(0,4):
        n = [p, q]
        c = [a[j], a[j+1]]
        cr = chinese_remainder(n, c)
        if (smallest<0):
            smallest = cr
        elif(cr < smallest):
            smallest = cr
        j +=2

    return smallest
```

Listing 4: Python function showing the process of creating private-keys from public-keys as per the Fiat–Shamir cryptosystem.

D Tonelli-Shanks Algorithm

This is an algorithm developed by Daniel Shanks in 1973 [69] which has roots to the work of Alberto Tonelli in 1891. This is an algorithm to solve an equation of the following kind (where p is a prime):

$$x^2 \equiv n \mod p$$

Algorithm 1: Finding square roots modulo x prime p - [70]

```
1. Set $k = n, z = u^q, x = a^{(q+1)/2}, b = a^q$.
2. Let $m$ be the least integer with $b^{2^m} \equiv 1 \mod p$.
3. Set $t = z^{2^{k-m-1}}, z = t^2, b = bz, x = xt$.
4. If $b = 1$, stop and return $x$, otherwise set $k = m$ and
   go to step 1.
```

E Example of Authentication

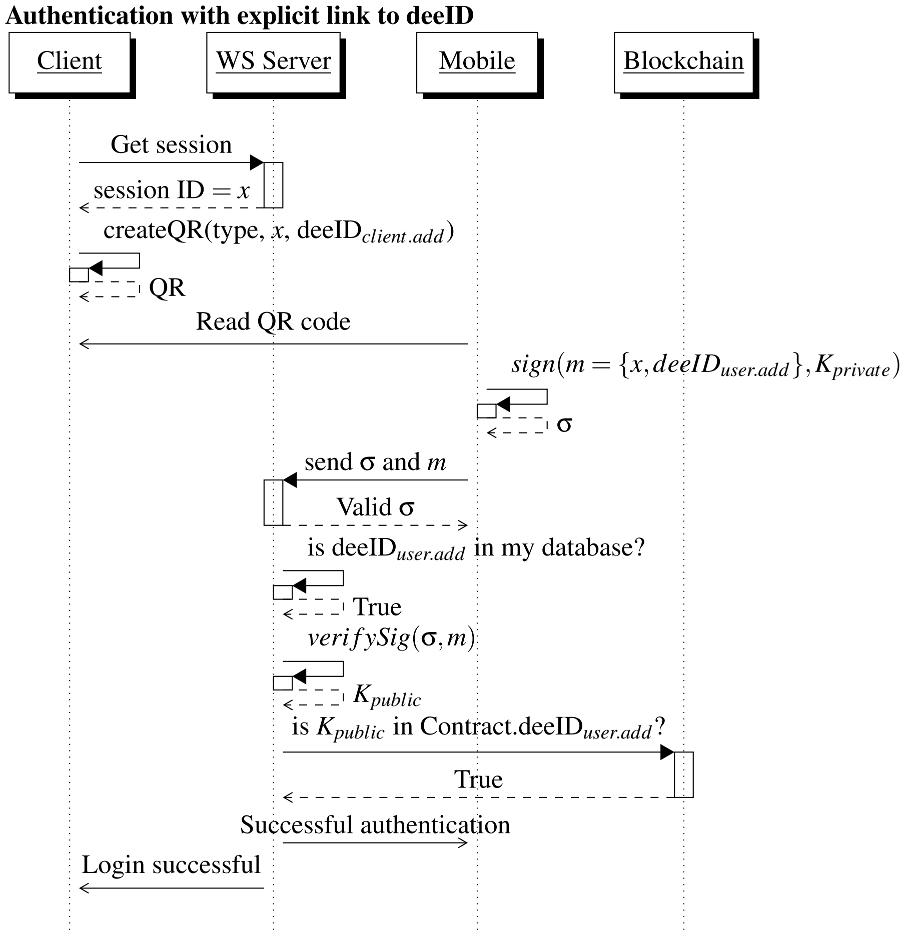

Authentication with explicit link to deeID
Client
WS Server
Mobile
Blockchain
Get session
session ID $= x$
createQR(type, x, deeID$_{client.add}$)
QR
Read QR code
$sign(m = \{x, deeID_{user.add}\}, K_{private})$
σ
send σ and m
Valid σ
is deeID$_{user.add}$ in my database?
True
$verifySig(\sigma, m)$
K_{public}
is K_{public} in Contract.deeID$_{user.add}$?
True
Successful authentication
Login successful

Figure 6: Sequence of authentication and the steps taken in order to authenticate whilst using deeID with one specific public-key.