Towards encrypting industrial data on public distributed networks

Preece, Joseph; Easton, John

DOI:
10.1109/BigData.2018.8622246

Towards Encrypting Industrial Data on Public Distributed Networks

J. D. Preece
Birmingham Centre for Railway Research and Education
University of Birmingham
Birmingham, United Kingdom
jdp225@student.bham.ac.uk

J. M. Easton
Birmingham Centre for Railway Research and Education
University of Birmingham
Birmingham, United Kingdom
j.m.easton@bham.ac.uk

***Abstract*—This paper addresses the problem of uploading large quantities of sensitive industrial data to a public distributed network by proposing a new framework. The framework combines the existing technologies of the distributed web and distributed ledger to provide a mechanism of encrypting data and choosing whom to share the data with. The framework is designed to work with existing platforms; the InterPlanetary File System (IPFS) and the Ethereum blockchain platforms are used as examples within this paper, though it is stated that similar platforms are capable of providing the requirements for the framework to operate. The framework uses the concept of the Diffie-Hellman Key Exchange (DHKE), and is implemented in three different mechanisms of the DHKE: one-step Elliptical-Curve Diffie-Hellman Key Exchange (ECDH); two-step ECDH; and Supersingular Isogeny Diffie-Hellman Key Exchange (SIDH). The paper discusses the security of each along with individual advantages and disadvantages, and concludes that the SIDH is the most appropriate implementation for future use due to it being post-quantum secure.**

***Index Terms*—distribution, networks, blockchain, cryptography**

I. Introduction

Since the advent of the World Wide Web (WWW) in the early 1990s, the vast majority of internet users have adhered to the transport protocols supported by World Wide Web Corporation (W3C). Whilst this performs to a standard that most users regard as adequate, the network architectures that the WWW rely on are outdated and are in desperate need of modernisation. Instead of the reliance upon centralised servers, information should be distributed across the nodes of the Internet to make storage of data cheaper and faster to access. In an industrial context, where the costs associated with storing data are a major concern, the higher bandwidth costs of centralised networks can become an issue. For example, using the distributed approach to streaming a video can save up to 60% in bandwidth costs over traditional architecture. [1]

However, with this distributed approach comes the problem of sharing sensitive data; the nature of a distributed web means that data on the network is visible to anybody who wishes to access it. Encrypting the data prior to uploading it to the distributed network is crucial to maintaining the secrecy of any sensitive data.

Self-encryption methods are straightforward; one encrypts the file with a symmetric key which can be stored locally at the owner's node, and then uses the symmetric key to decrypt the file when required. The symmetric key is never available in a public domain, and remains accessible only to the user. Instead, a method that allows users to encrypt data and share with selected individuals must exist.

This paper proposes a framework in order to achieve encryption of data whilst enabling the uploader to decide whom they wish this data is accessible by. By using a distributed ledger platform alongside the distributed web to record transaction history, provide identity and signatures of the users, and enable the safe exchange of symmetric keys through an insecure domain, a robust platform of provenance and shared responsibility is created, removing the need for centralised servers.

II. Background

A. Distributed Networks

A computing network can be described as a "collection of interlinked nodes that exchange information". [2] How said information is exchanged is governed by the architecture of the network, of which many varying models exist. Baran defines three such models, as illustrated in Figure 1; centralised, decentralised, and distributed.

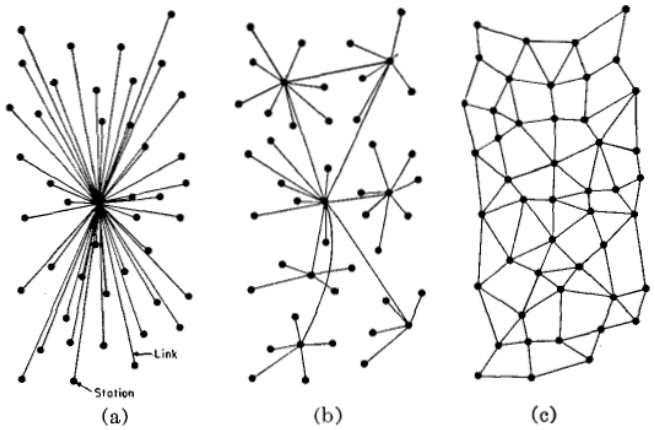

Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

Fig. 1.

Traditionally, the Internet relies upon centralised servers that act as a mandatory points-of-access. One aspect of the centralised Internet is the ability to regulate the network, as traffic must pass through mandatory servers before reaching the client. This is both advantageous and disadvantageous in the eyes of different users of the Internet.

Larger networks such as the Internet are made up of smaller sub-networks. These sub-networks can use any model. Individuals who host small websites will opt to run a single centralised server that handles all traffic and data. Large companies such as Google will use a distributed approach, hosting multiple servers and hubs in a number of different locations. This is so that data is not prone to loss due to a single point-of-failure. Note that although the sub-network is decentralised, the sub-network within the main network is centralised, as access is restricted and maintained by Google alone.

The distributed web is a term associated with transforming the WWW from a centralised model to a distributed model. This means that instead of accessing information stored at centralised servers, the information is distributed efficiently at every node on the network. Any node accessing information can then check locally or access remotely by locating a node that has the information. Because the information is distributed, there is no point-of-failure, and information can be accessed even if the information's source node is offline.

One notable attempt to achieve the distributed web is the IPFS. [4] The IPFS is an open-source peer-to-peer distributed file system, where each node on the network stores content it holds an interest in, along with indexing information to make tracking the location of other files easier. Each node is identified by a hash of a public key, where the key is generated by S/Kademlia static cryptographic puzzle and then stored in the distributed hash table. The network can make use of existing transport protocols such as Hypertext Transport Protocol (HTTP), Web Real-Time-Communication (WebRTC), and Micro Transport Protocol (μTP). A new routing system is introduced that combines the technologies of Coral and Kadmelia to maintain information to help locate information on the network, both locally and remote. Furthermore, a new block exchange protocol is introduced to govern efficient block distribution across the network. The file storage on the IPFS is analogous to Git. An object on the IPFS can store up to 256kb of data, and links to other objects. For files larger than 256kb, the file is broken down into these 256kb objects and linked via a Merkle Directed Acyclic Graph (MDAG). Though files are immutable and permanent, the IPFS supports version control and has a mutable naming system.

There are advantages that the distributed web (and implementations such as the IPFS) provide. One major advantage is the reduction of bandwidth. Assume a video is being streamed from YouTube. This is currently performed by requesting the video from the YouTube server each time; the same data is transferred between the server and the client every time the video is accessed. With the distributed web approach, the video is stored locally after accessing it once, removing the need to use bandwidth to stream the video further. It is estimated the distributed approach could reduce bandwidth costs by up to 60%.

B. Methods of Key Exchange

One of the earliest methods of key exchange is the DHKE, proposed in 1976. The DHKE establishes a shared secret between two parties by utilising the mathematical properties of Public Key Encryption (PKE). Each party may establish the secret independently of one another, removing the need to send sensitive data (i.e. a symmetric) across a network. The process is illustrated using a colour analogy in Figure 2. Alice and Bob

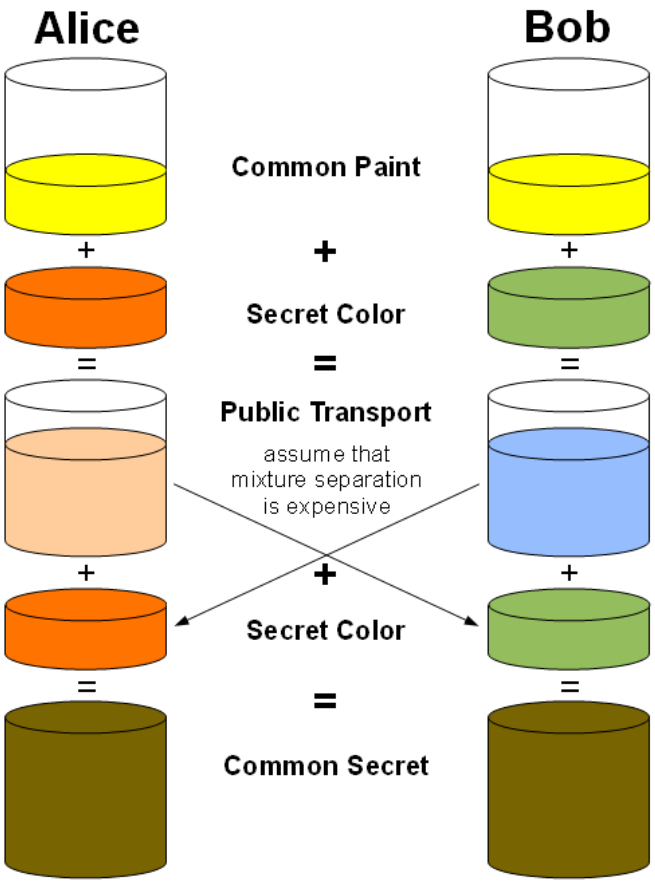

Fig. 2. The DHKE method, using paint mixing as an analogy

wish to produce a common colour, though wish to keep one colour hidden from the other. To achieve this, they agree on a common paint colour and mix in their secret colours. The new colours are swapped, and their secret colours are then mixed in. By doing this, they both end up with the common secret, without having to share their secret colours. Because paint separation is an expensive process, the secret colours cannot be derived from any mixtures. Eve - an adversary watching the process - only has access to the common paint and initial mixtures, meaning she cannot establish the common secret or either of the secret colours.

There are a number of different mechanisms that make use of the underlying principles of the DHKE. Three particular

examples used for implementation of this framework are ECDH, SIDH, and Supersingular Isogeny Key Encapsulation (SIKE).

1) Elliptical-Curve Diffie-Hellman Key Exchange: The ECDH is a modification of the original DHKE which makes use of Elliptical Curve Cryptography (ECC). Assume Alice and Bob have elliptical curve key-pairs (Q_A, d_A) and (Q_B, d_B) respectively, where d is a secret key and Q is a public key. A public key is represented by a point on the elliptical curve

$$Q = dG$$

where Q is the point on the curve, d is the secret key associated, and G is the generator of the cyclic subgroup. The coordinate on the elliptical curve is calculated as

$$(x, y) = d_A Q_B = d_B Q_A$$

where Alice calculates $d_A Q_B$, and Bob calculates $d_B Q_A$. The x coordinate is used as the shared secret. Because

$$d_A Q_B = d_A d_B G$$

and

$$d_B Q_A = d_B d_A G$$

it is trivial to see that the shared secret is equal for both parties.

The ECDH is compatible with keypairs generated by the Elliptical Curve Digital Signature Algorithm (ECDSA), and as such, any distributed ledger platform that utilises the ECDSA such as Ethereum.

Shor's algorithm is a hypothetically possible quantum algorithm that computes discrete logarithms to break the underlying ECC, with an estimate of 2330 qubits and 126 Toffoli gates required to break a 128-bit level of security. [5]

2) Supersingular Isogeny Diffie-Hellman Key Exchange: SIDH is a solution that modifies the methodology to secure the DHKE against post-quantum attacks such as Shor's algorithm. [6] As with the ECDH, certain parameters are made public prior to use: a prime p; a supersingular elliptical curve E; fixed elliptical points (P_A, Q_A) and (P_B, Q_B); and the orders of the points $(\omega_A)^{e_A}$ and $(\omega_B)^{e_B}$.

Alice begins by establishing the kernal of her isogeny by creating a random point

$$R_A = m_A P_A + n_A Q_A$$

where n_A, m_A are two random integers and $n_A, m_A < (\omega_A)^{e_A}$. Alice uses R_A to create an isogeny mapping ϕ_A, where the mapping is $E \to E_A$. Alice applies the mapping to P_B and Q_B to form points $\phi_A(P_B)$ and $\phi_A(Q_B)$ on the curve E_A. Alice sends E_A, $\phi_A(P_B)$, and $\phi_A(Q_B)$ to Bob. Bob then mirrors the steps made by Alice and sends her E_B, $\phi_B(P_A)$, and $\phi_B(Q_A)$.

Alice then forms

$$S_{BA} = m_A(\phi_B(P_A)) + n_A(\phi_B(Q_A))$$

and creates an isogeny mapping ψ_{BA}, where the mapping is $E \to E_{BA}$. Alice then computes the j-invariant

$$K = j(E_{BA}) = 1728\frac{4a^3}{4a^3 + 27b^2}$$

by using the Weierstrass equation. Bob mirrors Alice's steps to also obtain E_{AB} to calculate the j-invariant. Both curves are guaranteed to have an identical j-invariant.

It is confirmed that SIDH has a classical security of $O(p^{1/4})$ [7, 8] and is hypothesised that the quantum security is $O(p^{1/6})$ [6]. This suggests that the security level is 128-bit if the p is a 768-bit prime number.

The SIDH requires public keys of 330 bytes, greater than that used by the ECDSA. As such, SIDH is not compatible with existing distributed ledger platforms that make use of ECDSA to generate the keypairs.

III. Methodology

In order to combat the issue of data privacy issues within distributed networks, a new framework is proposed that integrates both distributed web and distributed ledger technologies to enable secure key exchange and permanently accessible data. The framework operates over a series of steps: encrypting plain text data and placing it on the distributed web; establishing a shared secret with an intended recipient and encrypting the symmetric key used to encrypt the plain text data; sending the encrypted symmetric key through a transaction on the distributed ledger; decrypting the symmetric key with the shared secret; and decrypting the cipher text with the symmetric key. There are two agents at play in the framework: Alice, who wishes to commit and send data; and Bob, who wishes to receive data from Alice. Eve, who wishes to intercept and read the data without prior permission, is also considered.

The framework is implemented in three varying methods: one-step ECDH, two-step ECDH, and SIDH. This is in order to compare the security and efficiency of two varying exchange tools against one another. The details of all implementations are explained in the following sections respectively.

A. One-Step Elliptical-Curve Diffie-Hellman Key Exchange

This approach utilises the ECDH methodology by enabling the two parties to independently generate a shared secret K. A symmetric key is derived by passing the shared secret through the Hash-Based Message Authentication Code (HMAC) Key Derivation Function (KDF) (HKDF) which is then used to encrypt Alice's plaintext. We assume that Alice and Bob have generated keypairs (Q_A, d_A) and (Q_B, d_B) respectively, where Q is a point on the elliptical curve. The keypairs are generated by the ECDSA, and as such can be used as Ethereum addresses. The process is described in the following sequence:

1B Bob invokes a request transaction, asking Alice to share data with him

1A Alice computes Bob's public key Q_B from the transaction and computes the point on the elliptical curve $(x, y) = d_A Q_B$

2A Alice passes the shared secret x (the x coordinate of the computed elliptical curve point) through the HKDF to derive a robust symmetric key K

3A Alice encrypts plaintext data she wishes to share with Bob by using K with the Advanced Encryption Standard (AES).

4A Alice uploads her ciphertext on to the IPFS.

2B Bob derives K by mirroring Alice's steps between 2A-3A

3B Bob accesses the IPFS file and uses K to decrypt the ciphertext.

The disadvantage of this approach concerns the fact that there is one shared symmetric key per pair of Ethereum addresses. Should Alice wish to share a piece of data with 10 people, she will have to establish a shared symmetric key with each person and encrypt the data 10 times. This is a computationally demanding process, as each version of the ciphertext must be made available on the IPFS.

B. Two-Step Elliptical-Curve Diffie-Hellman Key Exchange

A variation of the ECDH may be used, where a symmetric key is randomly generated for each piece of data being shared, and a shared secret is generated between Alice and Bob via a temporary elliptical curve key-pair. As with the one-step implementation described previously, we assume that Alice and Bob have generated keypairs (Q_A, d_A) and (Q_B, d_B) respectively. The process is described in the following sequence:

1A Alice generates a 128-bit symmetric key K_1 and encrypts the plaintext

2A Alice uploads the ciphertext to the IPFS

1B Bob invokes a transaction to request K_1 to decrypt the ciphertext

3A Alice checks for any unresolved requests on the blockchain. She performs this by checking the request addresses against her address whitelist.

4A If Bob's address is on the whitelist, Alice computes his public key Q_B and then computes the point on the elliptical curve $(x, y) = d_A Q_B$

5A Alice passes the shared secret x (the x coordinate of the computed elliptical curve point) through the HKDF to derive a robust symmetric key K_2

6A Alice encrypts K_1 with the AES, using K_2 as the key

7A Alice invokes a transaction to send Bob the encrypted K_1

2B Bob derives K_2 by mirroring Alice's steps between 5A-6A and decrypts the encrypted K_1

3B Bob uses K_1 to decrypt the ciphertext

The advantage of this approach is that each piece of data uses a unique randomly generated symmetric key which is used to encrypt the data prior to any requests being made. If two users agree to share data, they establish a shared secret via their respective elliptical curve keypairs. This maintains one shared secret between the users, whilst protecting the original symmetric key should Alice wish to withhold the data from Bob.

C. Supersingular Isogeny Diffie-Hellman Key Exchange

A further use of the Diffie-Hellman approach is SIDH. Whilst the underlying method behind this approach is identical to that of the approach described in Section III-B, the SIDH approach is post-quantum secure, ensuring security against adversaries with quantum computing capabilities.

1A Alice generates a 128-bit symmetric key K_1 and encrypts the plaintext

2A Alice uploads the ciphertext to the IPFS

1B Bob invokes an Ethereum transaction to request K_1 to decrypt the ciphertext. Within this transaction, Bob provides E_B, $\phi_B(P_A)$, and $\phi_B(Q_A)$

3A Alice checks for any unresolved requests on the ledger. She performs this by checking the request addresses against her address whitelist.

4A If Bob's address is on the whitelist, Alice computes his public key P_B and then computes E_A, $\phi_A(P_B)$, and $\phi_A(Q_B)$ to form an elliptical curve E_{BA} isogenous to the original curve E

5A Alice computes K_2, where K_2 is the $j - \text{invariant}(j_{AB})$ of E_{AB}

6A Alice passes K_2 through through the HKDF to derive a robust symmetric key $K_{\text{sym},2}$

7A Alice encrypts K_1 with the AES, using K_2 as the key

8A Alice invokes a transaction to send Bob the encrypted $K_{\text{sym},1}$

2B Bob derives K_2 by mirroring Alice's steps between 5A-7A and decrypts the encrypted K_1

3B Bob uses K_1 to decrypt the ciphertext

Because SIDH generates keypairs that are not compatible with ECDSA, any distributed ledger platform that uses the ECDSA is incompatible with this method. As such, a new distributed ledger platform that utilises a 330-byte public key must be implemented in order to achieve the same framework mechanics of the previous two implementations.

IV. Discussion

Whilst there are particular advantages and disadvantages to each variation of implementation, some aspects hold more significance than others. The underlying framework methodology - regardless of implementation - also has both advantages and disadvantages.

By using a blockchain platform, the ECC key-pairs can be used in two ways; shared secret generation for encryption of data, and digital signatures to ensure provenance and identity. Existing platforms that use the ECDSA and smart contracts, such as the Ethereum platform, can be used with the one-step and two-step ECDH implementations.

The application of the framework this paper has analysed is built with two existing technologies; the IPFS and Ethereum. Neither technology required modification for the framework to deliver the intended result. Hypothetically, any similar technologies can be used in place. For example, any blockchain platform that utilises ECC and supports smart contracts can

be used in place of the Ethereum platform. Furthermore, any distributed web platform can be used in place of the IPFS.

Once data is uploaded to the IPFS, it remains accessible for as long as the network is running. Deletion of data is impossible, unless all nodes on the network agreed to remove all local copies; something that is infeasible to accomplish if members of the network do not trust one another. Should a breach of the encryption occur, or the symmetric key to become public, or the Ethereum account of a recipient, it becomes possible for the data to be accessed to anybody.

The SIDH implementations provide security against post-quantum attacks. Due to the permanence of data on a public distributed network, this is an increasingly important consideration. Though quantum attacks are only hypothetical, quantum computing technology continues to make rapid progress and will one day pose a threat to the classical fields of cryptography that are so common with the centralised web.

One of the major disadvantages with this framework is the requirement of offline computation to perform encryption tasks. Due to the nature of the blockchain, all processes performed within a Solidity smart contract are visible to anybody who wishes to see them; even private variables may be accessed. For security purposes, it is strongly advised to avoid placing any sensitive data into a smart contract i.e. a unencrypted symmetric key.

The framework offers a lightweight solution to distributing industrial data between stakeholders over trustless networks. This is important to achieve in order to remove the financial and logistical burdens of centralised networks. One particular industry that could make use of such a framework is the railway industry in the United Kingdom. With a highly privatised railway industry, there are many competing companies which leads to siloation; the segregation of whole Information Technology (IT) systems into silos. This becomes a problem when a company may be willing to share some data; an entirely new piece of software must be issued in order to extract the data of interest, whilst maintaining protection over the data that it wishes to keep secret.

By storing data on a distributed network, the storage of data will be shared amongst the stakeholders and will make data accessibility easier. The framework will allow stakeholders to choose which data to share and whom to share it with. An example of where the framework could be applied is sensor data; the stakeholder may only wish for whoever is responsible for repairing particular aspects of the railway to see the relevant data; thus they can establish a secure bond of trust via the blockchain public keys whilst not having to worry about where the data is stored on the network.

V. Conclusion

Taking all security and performance aspects into consideration, it is predicted that the SIDH implementation of the framework will be most suitable for use. This enables users to achieve the original purpose of the framework - to encrypt data in a public domain and share with selected individuals - whilst providing post-quantum security. This is dependent upon results which are still being collected upon submission of this paper. Furthermore, this will be dependent upon years of use and scrutiny of SIDH in order to ensure it is secure.

Nonetheless, a step towards moving industrial data away from a centralised context has been theorised and implemented. With the forthcoming years of confirming SIDH's security, there are aspects of the framework which can be refined and improved. Looking to the future, there are a number of areas of research identified from the results of the framework discussed in this paper. One area of concern is computation of symmetric keys, and whether this can be shifted into a distributed context. This would alleviate the need to compute the symmetric keys offline, resulting in a framework where users can request and access data with no need for the data's creator to be online. This research will look into modifications of the virtual machines and script languages of the blockchain platforms in use.

Furthermore, research must be undertaken into how such a framework can integrate with data processing. As of this paper, the framework has the ability to store sensitive documents in a public domain, but all other manipulation of the documents must be performed offline.

References

[1] K. Nguyen and T. Nguyen and Y. Kovchegov. A P2P Video Delivery Network. Oregon State University. 2010.

[2] Institute of Network Cultures. Beyond Distributed and Decentralized: What is a Federated Network? http://networkcultures.org/unlikeus/resources/articles/what-is-a-federated-network/. [Online; accessed 17-Sep-2018].

[3] P. Baran. *On Distributed Commincations*. Research Memorandum. Rand Corporation, Santa Monica, California, 1964.

[4] J. Benet. IPFS - Content Addressed, Versioned, P2P File System. Protocol Labs. 2014.

[5] M. Roetteler, M. Naehrig, K. Svore, and K. Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. Cornell University. 2017.

[6] D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, 4th International Workshop, Taipei, 2011.

[7] J. Biasse, D. Jao, and A. Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves. University of Waterloo. 2016.

[8] S. Galbraith, C. Petit, B. Shani, and Y. Bo Ti. On the security of supersingular isogeny cryptosystems. University of Waterloo. 2016.