Anonymous Data Sharing Between Organisations with Elliptic Curve Cryptography

Mark A. Will and Ryan K. L. Ko
Cyber Security Lab
The University of Waikato
Hamilton, New Zealand
Email: {mark.will, ryan.ko}@waikato.ac.nz

Silvino J. Schlickmann
Innovation Directorate
INTERPOL Global Complex for
Innovation (IGCI), Singapore
Email: s.schlickmannjunior@interpol.int

***Abstract*—Promoting data sharing between organisations is challenging, without the added concerns over having actions traced. Even with encrypted search capabilities, the entities digital location and downloaded information can be traced, leaking information to the hosting organisation. This is a problem for law enforcement and government agencies, where any information leakage is not acceptable, especially for investigations. Anonymous routing is a technique to stop a host learning which agency is accessing information. Many related works for anonymous routing have been proposed, but are designed for Internet traffic, and are over complicated for internal usage. A streaming design for circuit creation is proposed using elliptic curve cryptography. Allowing for a simple anonymous routing solution, which provides fast performance with source and destination anonymity to other organisations.**

***Index Terms*—Anonymous Routing; Elliptic Curve Cryptography; Light-weight; Data Sharing; Secure Processing;**

I. Introduction

Since the first proposal of an anonymous network by David Chaum in 1981 [1], much research has been proposed to further improve the concept either through greater anonymity [2][3][4], or faster performance [5][6][7]. However focus has been on Internet users and traffic. This paper presents the issue of an organisation accessing information within a closed group under a data sharing agreement, using a closed anonymous network to hide their actions to others. The concept of a closed network is not unique, but to the best of our knowledge the use case of data sharing between organisations or agencies has not been explored. Furthermore with the increasing usage of secure/private processing, data is already encrypted before leaving the client, but there is still a requirement for the source to be hidden.

The ability to create an internal anonymous routing network will facilitate data sharing between organisations, while keeping their actions private. Often search history can leak information about the searcher and the project they are researching. For example Google uses your search history for advertisement purposes with AdSense. In Section II some domain specific use cases are given. The primary use case of this paper is to allow an entity to host information, where other entities can search this data without the hosting entity itself gaining any information. When combined with encrypted search operations, will keep the other organisations actions private, along with their queries.

Many related works in Section III are designed for an anonymised network for Internet traffic, whereas this paper is addressing the issue of providing an anonymised network for an internal organisation or group of organisations. Therefore the scope of whom a source needs to remain anonymous from is reduced to only the other peers. Networks (for example Internet service providers) in-between these peers cannot see the request or response data as it is encrypted, but could try infer whom the source and destination by traffic analysis, (discussed in Section IX-G).

Elliptic Curve Cryptography (ECC) is proposed in parallel with Advanced Encryption Standard (AES) as a method of providing public key support, as well as smaller key and cipher text sizes, with a simplified anonymous routing protocol presented in Section IV. A brief attack analysis in Section V shows that the protocol can hide the source and destination, while protecting the request and response data. Simulations of performance are given in Section VII, and shows the proposed circuit creation approach is acceptable even for small files. The paper is concluded in Sections VIII, IX and X with comparisons between an closed anonymous network and an Internet-based one, as well as current open questions and potential solutions.

II. Use Cases

A. Law Enforcement and Government Agencies

A challenge for law enforcement agencies is keeping their investigations private, but having the ability to share information. INTERPOL helps to facilitate this collaboration between its member countries [8], however there is still the issue of trust and privacy around information sharing. For example an entity hosts information regarding cryptocurrency, then ideally the other entities can search, view and download this information, without revealing to the host which entity is accessing it. A distributed example is where a country has a fingerprint of a person of interest [9]. They can query against other countries databases, without revealing which country the request came from.

With law enforcement entities, there is the expectation that they should behave ethically and to the law. The same could be said of government agencies, but there is a growing threat of insider attacks [10]; potential information leakage to the press invalidity cases [11], massive operational leakages [12],

and leaked terrorism information [13] are a few examples in recent years. Even accidental data leakages occur when sharing or transferring information, with previous Europol investigations and sensitive data leaked online without the users knowledge [14].

Therefore data leakages are a problem for law enforcement and government agencies, which has a negative impact on information sharing between organisations. Outright data leakage is still an issue, but this paper aims to hide their activities to provide better control over data sharing, secure data transfers, and stopping the ability to learn what others are up to. Keeping law enforcement activities secret (for example investigations) is critical, and helps to increase the trust in accessing shared information. Combined with encrypted data processing, enough privacy can be achieved for some countries strict legislation around data sharing [15].

B. Healthcare and Finance

Research for new medicines and general health related issues should be an area where data sharing and collaboration is critical to saving lives and improving wellbeing [16][17]. However often these sectors can be driven by profit and discovering the next super drug, that they are reluctant to share data. Data sharing should also be more active between hospitals, general practices, and other expert organisations [18]. Patient privacy is a challenge for data sharing in the healthcare sector, even with anonymous records [17]. Note there is the challenge of authenticating the release of patient information, but this is out of scope.

The sharing of financial information could show vulnerability or expose a banking systems flaws. Furthermore searching for links between transfers for detecting fraud and other illegal activity could be made easier with data sharing. For example countries have pledged to share tax information with each other for greater trust and transparency in the banking sector [19]. There may arise the need for countries to remain anonymous while searching this shared data.

C. Encrypted Processing

Applications or services offering private or encrypted processing [20][21][22], such as searching [23][24][25], aim to protect the privacy of users data while in-flight. However information can be leaked from who and where the request came from. For example if a user is searching a document while connecting through a supermarket's access point, one could guess the document is a shopping list. For organisations accessing shared data, the fact that they are requesting or searching some data could be revealing, even if the search is encrypted. Applications that are challenging to implement securely (with homomorphic encryption for example), anonymising requests would add a layer of security with little effort. Therefore the ability to anonymise who and where a request originated from is an important aspect of private processing.

III. Related Work

A. P2P Designs

Both Tarzan [26] and MorphMix [27] are designed such that each peer is a potential relay or originator of traffic. This hides the origins of a request to the other peers and final destination. A limitation of Tarzan is that a node only has a small subset of other nodes [28]. For organisations each hosting information, each node should have knowledge of all others. With MorphMix the source does not chose the route through the network, instead it is done by intermediate nodes. Crowds [29] is also a peer-to-peer (P2P) model but does not include public-key cryptography. The model of a P2P network is ideal for data sharing between a large number of organisations, where each can be a peer in the network.

B. High Latency

Babel [2], Mixmaster [3] and Mixminion [4] introduce large latencies for achieving anonymity. For example the Mixmaster protocol defines that the remailer must collect several encrypted messages before sending the message it has just created [3]. For an internal network between organisation, this could be problematic if there is a limited amount of traffic being generated. Even though these protocols give better anonymity than low latency and lightweight protocols, for the use cases presented in Section II the introduced delay and small amount of traffic would limit their usefulness.

C. Tor

The anonymous onion routing protocol Tor [30], uses a small number of relays to hide the origins of traffic in the Internet. Some limitations of Tor are the setup time for circuits and managing the directory for relay nodes. There has been many related works on speeding up circuit creation [31][32][33]. However these still have high overhead and few hops when compared to lightweight designs.

D. Lightweight

Wireless networks with limited resources require simpler designs and fast circuit creation, but often avoid public key cryptography because of computational overhead [34]. Lightweight Anonymity and Privacy (LAP) proposed in [5] addresses the issue of only protecting against the end-server with low latency and easy circuit creation. Limitations include no use of public key cryptography or mention symmetric key exchange, and are designed for Internet communications. Improvements are proposed [6][7], but can be simplified for an internal network between organisations.

E. AES Encryption

Rijndael or as it is commonly known, Advanced Encryption Standard (AES), is a method of data encryption with symmetric keys [35]. The advantage over other cryptography schemes for layered encryption is the cipher-text is the same size as the plain-text. With inputs larger than the key size, they are split into blocks and chained together. The core operations of AES are exclusive or (XOR) and bit rotating, giving it

good performance as well as small cipher text sizes. The two components required for encryption and decryption are the key (256-bit for example), and initialisation vector (IV).

F. Elliptic Curve Cryptography

A public key cryptosystem, Elliptic Curve Cryptography (ECC) was proposed independently by Neal Koblitz and Victor Miller in 1985, and its cryptographic strength comes from the elliptic curve discrete logarithm problem being hard [36]. The advantages of ECC are smaller key sizes, smaller cipher text sizes (less data transferred) and faster computation times [37][38] when compared with non-ECC schemes such as RSA. For example the *secp256r1/nistp256* curve (256-bit) is comparable to the cryptographic strength of a RSA 3072-bit key [39].

IV. Simplified Protocol Design

A. Peer Information

Each node will have an IP address, ID, and public encryption key for every other node, where the ID is used as a next hop identifier. With a closed network, all peer information can be shared via secure mediums (out of scope of this paper, but is discussed in [40]) as a node is added. Once setup, nodes will not be added or removed often, hence a closed network. This mitigates the need for directory servers and the ability for nodes to automatically join the network. As the number of nodes will be relatively small compared to Internet onion routing protocols, each node can periodically ping others nodes to check for availability. However these nodes should be permanently alive, and not removed from the network without prior warning, other than an unexpected failure. The other benefit is that only verified nodes can join the network.

B. Circuit Construction

With each peer having the public keys of all other peers, the need of fully secure layered encryption is not required for data protection. However there is still a need to guarantee the correct path has been taken, which is where AES is used in Tor (as well as data protection). For simplicity, AES will not be used for providing the layering, instead a random value r will be XORed over all the data with chaining. Each hop and the destination receive a unique random value. The initial data is XORed with all r values, then each hop will XOR the r it receives, removing layers until the destination removes the final layer. Note that AES can be used, but the performance will decrease slightly and the circuit path headers will be larger.

The protocol was designed on top of TCP for packet ordering, guaranteed delivery and a form of corruption checking. In order to provide a new circuit for each TCP stream, circuit construction must be efficient otherwise the user will experience latency. Therefore the circuit definition is sent along with the data. Because each stream has its own circuit, the need of maintaining state on each node is removed. However if the circuit is required to be used multiple times, state can be maintained.

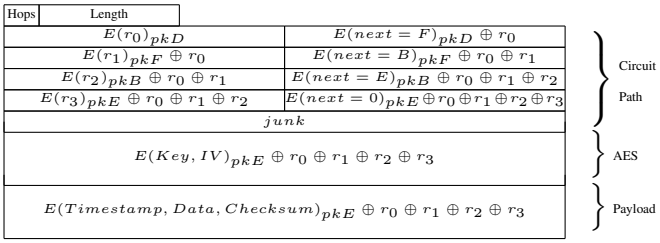

Fig. 1. Protocol Structure

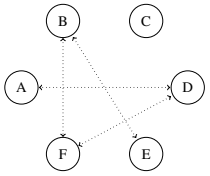

Fig. 2. An example circuit for A requesting data from E.

The circuit header consists of two values: (1) a random value r, and (2) the next hop identifier. These values are encrypted with the current hops public key. The cipher values for the next hop needs layering removed before decryption, hence it receives the cipher value of r first. They are removed from the stream when the data is forwarded to the next hop. To prevent a node guessing where it is in a circuit, when a node removes its r value and next hop value, the node adds extra rubbish at the end of the circuit path [31].

An example of the stream structure is given in Figure 1, using the example circuit in Figure 2. The first two pieces of information are the maximum number of hops, and the length of the overall stream. Number of hops is needed in order to add rubbish (in this case 5), and length is used so that the stream can be closed when it is finished. If the stream were to remain open, the length could possibly be removed. The circuit path comes after with each hops r value, and the next hop information. A node can only decrypt the next information after it decrypts r, as the cipher value is obfuscated with that value. Therefore the previous hop only has the cipher value for the next r value, as the remaining data is obfuscated.

When the stream reaches the destination, the AES key and IV are decrypted. The payload includes the request data, along with a timestamp and checksum value. The timestamp is important to stop replay attacks, as discussed in Section V. The response is encrypted with the AES key using the IV provided in the request. The cipher text is then obfuscated with the destinations r value. As the response is streamed back through the circuit, each node obfuscates further – adding more layers – until received at the source. The source node can then remove each layer since it generated the r values, before decrypting the response.

C. Removing Circuits: Direct Path for Data

Many related protocols have been designed to try and hide the user from as many entities as possible. However with the proposed use cases, the organisations only need to hide their actions from the other peers. Therefore instead of sending a

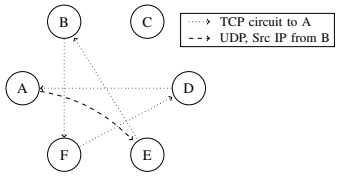

Fig. 3. Only the response is using the circuit, to reduce latency for the encrypted request.

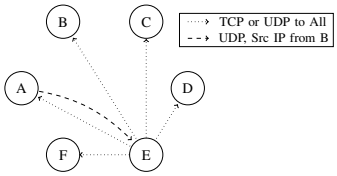

Fig. 4. Broadcast the encrypted response instead of using a circuit.

request through the circuit, it could be sent directly to the destination, with the encrypted path back and the source IP address set to the first node (on the way back) as shown in Figure 3. This initial request would need to use UDP, then the downloading of data using the anonymous circuit can use TCP to keep packets in order and handle dropped packets.

The other option is to remove the circuit altogether and broadcast the response to all nodes. An example is given in Figure 4. This has a few disadvantages; large files will have N download streams compared to 1, a large number of nodes will consume a large amount of bandwidth, and each node knows that a request was sent to that destination. However for small files this could offer a large performance increase. The same protocol can be used, minus the next node information. Therefore the client still generates a key for the destination to encrypt the response.

The limitation of these two approaches is that the source request, even with the changed source IP address, still originates from the source node. Networks (for example Internet service providers) in-between these peers could try infer whom the source and destination is by traffic analysis. However the scope is to hide the actions from the other peers, not every entity in the Internet as all data is encrypted. Therefore the two speed-up approaches meet this requirement.

V. Attack Analysis

A. Breaking the Request

In the case where the destination is known, for example a single organisation is hosting data for others to search and use, the destination public key will be known. Also the second to last hop also knows it is the last hop before the destination as the next ID will be that of the destination (as there is only one in this example). However it still does not know the source node, and what data/request is being sent to the destination.

One attack vector to solve the request would be for the second to last node to change the AES key and IV by encrypting new ones with the destinations public key. Therefore when the data from the destination is encrypted using an AES key the node knowns. However the XOR chaining value is not known. If the node tried to change the XOR value so that the AES key gets decoded correctly, the request data would then become incorrect. The response from the destination in this case would be a message saying the request is corrupt. But this message cannot be encoded using XOR values as this would reveal the XOR value used by the destination to the second to last node (first node on path back). Therefore the XOR value needs to be guessed when changing the AES key, which has 2^{32} possibilities. The node could try all options, but this large number of corrupt requests would be detected by the destination.

The destination will not know which peer is corrupting the requests, as it could be one of the other peers in the circuit. Therefore blocking a peer for corrupting traffic is difficult. With a limitation of 100 request per second, the destination cannot process 2^{32} requests a year, meaning guessing the XOR value is very time consuming. Even if the value is guessed, only the response to the request would be known (after changing the AES key). Finally if the request includes a timestamp, the requests XOR value would have to be guessed within a few seconds. Otherwise the destination can either ignore the request or send a timeout message like the corrupt message.

When the destination is unknown, it is difficult for a node to try and change request data as the public key is unknown, and the XOR values for the other nodes are not known. Therefore if it is unfeasible for a node to break a request when the destination is known, it is more unfeasible when the destination could be any of the other nodes.

B. Changing the Response

The other attack vector is where a node tries to modify the response data. Firstly the response data is encrypted with AES where the node does not known the key or IV used, and the XOR chaining means any change to the path or data will corrupt it.

C. Denial of Service

This model could be abused to slow down another organisation in the form of a Denial of Service (DoS) attack, especially if the broadcast response technique is adopted. However limitations such as number of permitted flows at one time can help. A DoS on the anonymous network would not be any different to an attack on a shared network without anonymisation, aside from the critical point being hit sooner.

D. Outside Attack

If an organisation is compromised, then the threat level of an attacker trying to analyse traffic in the anonymised network is not high. There are much bigger threats, such as phishing and data leakages. The proposed protocol is designed to stop organisations learning any information while sharing information. Also by only allowing certain operations and defining which data can be requested, should keep other organisations protected if another is compromised.

VI. Proof-of-Concept Implementation Details

Keeping with the philosophy of this paper, the implementation was initially created in Python for its simplicity. However the performance cost was large due to type packing and unpacking, so C was used instead.

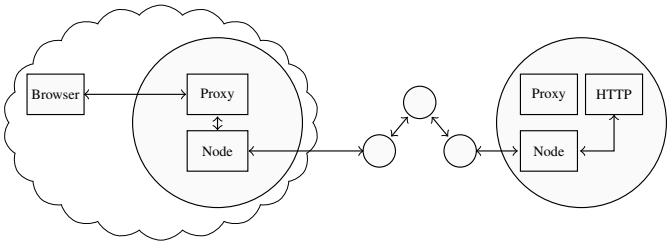

Fig. 5. Example of implementation process flow between browser, nodes, and service where a circle represents a node in the closed anonymous network. Note the browser could be on another machine, but on the same network within the organisation.

A. SECCURE

A toolset written in C by B. Poettering, SECCURE Elliptic Curve Crypto Utility for Reliable Encryption (SECCURE) has implementations of a selection of asymmetric algorithms based on ECC [41]. In particular the encryption/decryption functions were used within the node implementation with wrapper functions, and the toolset was used to generate ECC keys. The wrapper functions were *eccEncrypt*, *eccDecrypt*, *eccClean*, *eccInit*, which mimicked the *app_encrypt* and *app_decrypt* functions of SECCURE. The signing functions were not used, however in future work could be used to verify the response data.

The encryption of plaintext in SECCURE is not actually achieved with ECC, instead AES is used. However the AES key and initialization vector are generated by the encryptor, and encrypted with ECC. When the destination receives the cipher text, it first decrypts the key and initialization vector with its ECC private key, allowing it to then decrypt the plaintext message. This technique keeps the encryption of messages smaller than they would be with many other cryptography schemes, while maintaining a public/private key system.

B. Implementation Flow

The implementation follows the example in Figure 5, where a user from an organisation is requesting data from a HTTP server from another organisation. The browser can be within the organisations network, and sends the requests to the proxy. The proxy process then encrypts the request, defines a circuit and AES values, sending them to the node process. The proxy process could exist on each machine in the network, in order to encrypt the request before sending it to the node. The node processes this request as it would from another node – there is no difference between data received from another node and a proxy.

For each request, a new node process is spawned to handle that stream of data, including the response. This saves from needing to maintain state and stream or circuit identifiers. Because of the context of the use cases, where only a few streams would occur at once, having a seperate node process for each stream is acceptable. However in the context of an Internet onion router, this could be too costly. The state machine used for the node process is given in Figure 6, where $x = sizeof(E(r)) + sizeof(E(next))$ and $y = sizeof(E(aeskey)) + sizeof(E(aesiv))$. There are two

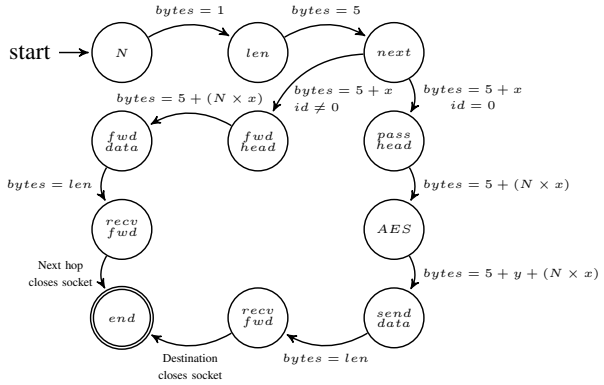

Fig. 6. State machine for a spawned node process.

distinct paths for when the stream has reached the destination, or needs to be forwarded. This is because when the data is being forwarded, the r value can be removed and then the data is sent. Also the $fwd\ head$ state needs to add rubbish to replace the circuit header it removed.

The transitioning in Figure 6 is given in terms of bytes received, but can be made simpler by changing the amount of data to read from the socket. A definition of each state is given below.

- N: reads the maximum number of hops.
- len: reads the length of the request.
- $next$: reads and decrypts the first circuit header, giving $r = Decrypt(cipher_r)$ and $id = Decrypt(cipher_{id} \oplus r)$
- $fwd\ head$: reads the remaining circuit headers, removing r, and adding junk to replace the removed header.
- $fwd\ data$: removes r before forwarding the data to the next hop. Note the data is streaming so the process does not need to wait for all the data to be received before forwarding it.
- $pass\ head$: consumes the junk circuit headers.
- AES: decrypts the AES $key = Decrypt(cipher_{key}) \oplus r$ and $IV = Decrypt(cipher_{IV}) \oplus r$.
- $send\ data$: forward the data after decrypting it to the local machines service (for example a HTTP server).
- $recv\ fwd$: both states perform the same function, and receive data from the destination/next hope, XOR with r and forward it back to the previous hop. However when $id = 0$, the data must be encrypted first.
- end: when all data has been received and forwarded, the node can clean up and terminate.

VII. Performance

All results were performed on a Late 2013 Mac Pro (*3.7GHz Intel Xeon E5, 16GB Memory*). With nodes on the same machine, OpenBSD's Packet Filter was used to add latency and limit bandwidth. The *SimpleHTTPServer* Python module was used for the web server. Note the implementation was proof-of-concept, and could be optimised further.

A. Circuit Creation

Testing circuit creation time is difficult given the circuit information and encrypted request data are in the same TCP

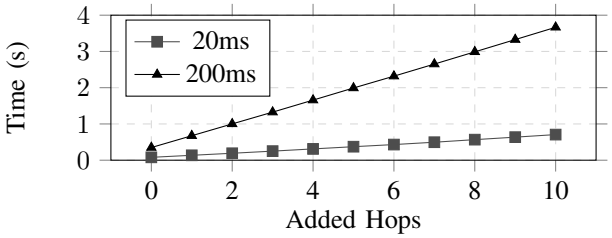

Fig. 7. Circuit creation time for varying number of nodes, for 20ms RTT and 200ms RTT between nodes.

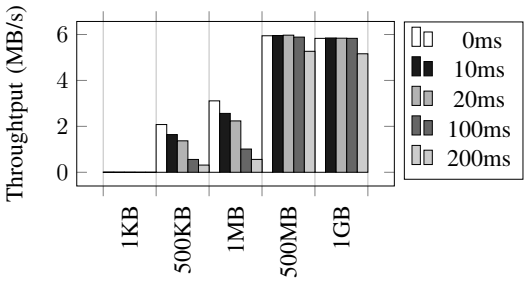

Fig. 8. Throughput versus file size for different latencies.

stream. With the nodes running on the same local machine, the timestamps would be accurate and can be used to measure circuit creation. Figure 7 shows the time difference between the first node generating and encrypting both a path and AES information, to when the circuit reaches the destination after the AES information is decrypted. The latency was 20ms or 200ms RTT between every node, and bandwidth was limited to 1Mbit/s. For comparison, Tor can take a few seconds to create a circuit with only 3 nodes [42].

B. Downloading Large Files

The time for a HTML file (8.4KBs) to download with 3 other nodes (5 in total) was on average 0.283s with a 20ms RTT time between each node in simulation. Figure 7 showed this was mostly circuit creating and stream time. Therefore the throughput for larger files was tested and is given in Figure 8 with varying RTT between nodes. The number of nodes was 5, where 3 nodes are not the source or destination. This shows that throughput is consistent once the overhead of circuit creation is reduced. However as the latency increases, the throughput will decrease. Given the forwarding is performed at the application layer, a throughput of > 5MB/s is respectable.

The throughput for the 1GB file without going through the anonymous network for 100ms RTT was 18.6MB/s. Note that this RTT was between the source and destination, and the anonymous network had 3 other nodes with this latency. The same download with a total latency of 400ms RTT only achieved a throughput of 4.7MB/s. One reason for this being slower than the anonymous network (5.8MB/s) with the same total latency is that the data/acknowledge packets take 200ms (one-way) to reach the destination/sender. Where with the anonymous network each acknowledge packet was sent between nodes, not the total distance.

Unlike networks such as Tor, the presented use cases in Section II will not exert a large number of parallel requests on the anonymous network. However the performance on multiple requests is more depend on available bandwidth and network queue sizes, than the processing power of the nodes.

VIII. Problems Avoided with a Closed Network

A. Keeping Peers Alive

A limitation with traditional peer-to-peer solutions is motivating peers to stay in the network once they have completed their tasks [30]. However with a closed network between organisations, the peers should always be up to allow their shared data to be accessed. The network peers should run on the same machines as services (HTTP, FTP), saving power and keeping requests secure.

B. Exit Nodes

With unencrypted services, the exit nodes of an anonymous routing network can see the traffic in plain-text. Even with encrypted services, the exit nodes could try to perform a man-in-the-middle attack, or other forms of attack specific for an exit node [43]. This is an advantage of having the web servers for example being part of the closed network with public key cryptography, such that even unencrypted services like HTTP or FTP are protected, and man-in-the-middle attacks are made more difficult.

C. Illegal Activity

With open anonymous routing networks, they are used to route both legal and illegal traffic. A study on the dark web showed that 57% of the active *.onion* addresses were illicit [44]. Most users of the Tor Browser never visit the dark web, however it still makes up for 3-6% of overall Tor traffic [44][45]. This type of activity is not something that is possible on a closed network of organisation peers, with limited operations for data sharing. This is because the destination node will only submit requests to services running on the local machine, in this case a web server.

D. Circuit Creation Time

As shown in Section VII-A, the time to create circuits is minimised. However given a new circuit is created each time, leads to an open question in Section IX-E. If circuits are only used for large file downloads, and the faster and simpler broadcast method is used for regular browsing, may not be a problem.

IX. Open Questions

A. Timezones

Depending on the number of organisations and peers, their physical location could be problematic as different timezones mean different working hours. For example an employee in New Zealand is probably not going to make a request at the same time as someone from England. Even with perfect anonymous routing, the time of the request leaks information on which organisation/peer could have submitted it. A simple solution would be to batch requests so that they all happen at the same time regardless of organisation, but this is not

ideal. However if enough organisations are within a similar timezone, only those nodes could be considered for the circuit, even though this is still limiting.

B. Network Latency for International Circuits

A problem which could have a similar solution to timezones, routing requests across the globe will have latency overheads. Assuming just adding latency and not limiting overall bandwidth, the downloading of large files or datasets was not greatly impacted by international peers, as shown in Section VII-B. However for searching and browsing information, this latency is going to be frustrating for a user. Good interface design can help hide latency and reduce the impact by being responsive, however this still could be a problem. For example a 8.4KB HTML file took 0.283s to download with a 20ms RTT between each node (5 in total). However the same time increased to 1.36s with a 200ms RTT between each node. Law enforcement agencies will need quick information on a case in certain situations. Possibly a "fast" option could be made available where only a few peers are used, but this should not be known to other peers.

C. The Element of Trust

Even with anonymous routed circuits, because each organisation has agreed upon sharing information and being apart of the group, there is an element of trust that each peer will behave in a trustworthy manner. For example a peer does not purposely corrupt circuits (which is an issue with other anonymous routing protocols as well). Note that the concept of using any anonymous routing protocol for organisations sharing data does not guarantee 100% privacy, but does help to provide more assurances.

D. Authentication

Providing authentication while keeping the source anonymous is an open problem. With the proposed anonymous network closed (where only permitted entities can join), this could be seen as a form of authentication. However this is assuming the public keys for the entities remain private to only the closed group.

E. Circuit Refresh

An open question in [30] is how often should a circuit be refreshed. This paper has discussed a method for more efficient circuit creating, with random paths that can include the client and destination. However creating a new circuit for each stream may lead to intersection and predecessor attacks [30][43]. Also by allowing a path to include a peer multiple times could allow for the two to be linked depending on the volume of traffic.

F. Linking TCP Streams

Another current problem with anonymous routing, and specifically ones which create circuits often is the ability to link streams together. When searching for information, the queries could be linked together at the destination. However the source should still remain unknown, and furthermore with encrypted search capabilities would prevent any linking based on queries. Another point is with limited overall requests, it could be assumed that queries close together are related.

For the nodes in the circuit to link data streams together would be difficult. However the point above could be applied where streams created in close proximity are related. For a large number of requests, if the source has 0% probability of being in the circuit path, a node could discover the source since it would never forward to that node.

G. Network Providers

Even though this is out of scope of the paper, network providers in-between the organisations could learn the source and destination for requests. Problems could arise for example if law enforcement starts requesting network traffic from an Internet service provider of another entity in the closed network. This could allow them to learn where a request of interest originated. Other issues with network providers is the potential to discover the r values used. But this can be solved by switching to AES for circuit verification as suggested in Section IV-B.

Attempts to hide the source and destination from network providers with the proposed protocol are challenging, as well as for other protocols. In order to hide the destination, the destination could create a random stream that has the same properties of the request, with another destination chosen at random. The request data is replaced with dummy data, as well as a size for the response data. The fake destination returns random data which reaches the real destination, and replaces it with the real response. Hiding the source is more challenging since the stream has to originate from somewhere. One technique could be for nodes to randomly create paths with random request data padded to a size of a large request. If a node wants to send a real request, it could wait until it receives a dummy request and change it. These techniques require more exploration in future work.

X. Conclusion

Where source and destination information need only be hidden from other peers in the closed network, a simplified anonymous routing protocol can be used for better performance. Elliptic curve cryptography was used in conjugation with AES to provide asymmetrical encryption for reduced cipher text sizes and faster encryption. Chaining of nodes with a simple XOR function is not sophisticated but as discussed in Section V, with data already encrypted, provides *good enough* security for an internal network. While AES can also be used to further improve the scheme by replacing r.

A closed anonymous network can also offer some level of authentication with is difficult for anonymous applications. With the increasing ability to create applications with encrypted search capabilities, anonymising the request is still an important step to achieving hidden usage. Future work includes combining the anonymous network with a secure searching application for law enforcement, edging closer to the goal of near perfect anonymity.

ACKNOWLEDGEMENTS

This research is supported by STRATUS (Security Technologies Returning Accountability, Trust and User-Centric Services in the Cloud) (https://stratus.org.nz), a science investment project funded by the New Zealand Ministry of Business, Innovation and Employment (MBIE).

REFERENCES

[1] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[2] J. Chroboczek, "The Babel Routing Protocol," *April*, 2011.

[3] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster Protocol Version 2," Online https://tools.ietf.org/html/draft-sassaman-mixmaster-03 (Accessed 03/03/17), December 2004.

[4] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 2–15.

[5] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and W. Meng, "Lap: Lightweight anonymity and privacy," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 506–520.

[6] J. Sankey and M. Wright, "Dovetail: Stronger anonymity in next-generation internet routing," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2014, pp. 283–303.

[7] C. Chen and A. Perrig, "Phi: Path-hidden lightweight anonymity protocol at network layer," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 100–117, 2017.

[8] INTERPOL, "Overview," Online https://www.interpol.int/About-INTERPOL/Overview (Accessed 16/02/17).

[9] E. Kelly, "The Agreement on Enhancing Cooperation in Preventing and Combating Crime Q & A," Online https://www.beehive.govt.nz/sites/all/files/The_Agreement_on_Enhancing_Cooperation_in_Preventing_and_Combating_Crime_FAQs.pdf (Accessed 06/03/2017).

[10] "Preventing Insider Fraud at Government Agencies," Intellinx Ltd., Tech. Rep., 2014.

[11] D. Purdum, "Billy walters files motion to dismiss federal insider-trading case," Online http://www.espn.com/chalk/story/_/id/18470705/billy-walters-files-motion-dismiss-insider-trading-case-alleging-government-misconduct (Accessed 22/02/17), January 2017.

[12] B. I. Paul Szoldra, "This is everything edward snowden revealed in just one year of unprecedented top-secret leaks." Online http://www.businessinsider.com/snowden-leaks-timeline-2016-9 (Accessed 22/02/17), 2016 September.

[13] L. Vaas, "Database of 2.2m suspected terrorists, money launderers leaked online," Online https://nakedsecurity.sophos.com/2016/07/01/database-of-2-2m-suspected-terrorists-money-launderers-leaked-online/ (Accessed 22/02/17), July 2016.

[14] B. Leo Kelion, "Secret Europol terror data found online," Online http://www.bbc.com/news/technology-38158258 (Accessed 22/02/17), November 2016.

[15] New South Wales Government, "Data Sharing (Government Sector) Act 2015 No 60," Online http://www.legislation.nsw.gov.au/#/view/act/2015/60/full (Accessed 14/03/17).

[16] B. Fabian, T. Ermakova, and P. Junghanns, "Collaborative and secure sharing of healthcare data in multi-clouds," *Information Systems*, vol. 48, pp. 132–150, 2015.

[17] Data Futures Partnership, "Exploring Social Licence - a conversation with New Zealanders about data sharing and use," Online http://datafutures.co.nz/assets/Uploads/DFP-Engagement-doc-FINAL.pdf (Accessed 06/03/17), July 2016.

[18] P. Groves, B. Kayyali, D. Knott, and S. V. Kuiken, "The'big data'revolution in healthcare: Accelerating value and innovation," 2016.

[19] M. Dakers, "51 countries sign up to share tax data," Online http://www.telegraph.co.uk/finance/newsbysector/banksandfinance/11196423/51-countries-sign-up-to-share-tax-data.html (Accessed 15/02/17), October 2014.

[20] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *STOC*, vol. 9, 2009, pp. 169–178.

[21] M. A. Will, B. Nicholson, M. Tiehuis, and R. K. Ko, "Secure Voting in the Cloud using Homomorphic Encryption and Mobile Agents," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*. IEEE, 2015, pp. 173–184.

[22] M. A. Will, R. K. Ko, and I. H. Witten, "Privacy Preserving Computation by Fragmenting Individual Bits and Distributing Gates," in *Trustcom/BigDataSE/ISPA, 2016 IEEE*. IEEE, 2016, pp. 900–908.

[23] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proceedings of the 29th conference on Information communications*. IEEE Press, 2010, pp. 441–445.

[24] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*. IEEE, 2011, pp. 383–392.

[25] M. A. Will, R. K. Ko, and I. H. Witten, "Bin Encoding: A User-Centric Secure Full-Text Searching Scheme for the Cloud," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1. IEEE, 2015, pp. 563–570.

[26] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 193–206.

[27] M. Rennhard and B. Plattner, "Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection," in *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*. ACM, 2002, pp. 91–102.

[28] G. Danezis and C. Diaz, "A survey of anonymous communication channels," Technical Report MSR-TR-2008-35, Microsoft Research, Tech. Rep., 2008.

[29] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM transactions on information and system security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[30] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.

[31] L. Øverlier and P. Syverson, "Improving efficiency and simplicity of tor circuit establishment and hidden services," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2007, pp. 134–152.

[32] R. Snader and N. Borisov, "A tune-up for tor: Improving security and performance in the tor network." in *ndss*, vol. 8, 2008, p. 127.

[33] C. Tang and I. Goldberg, "An improved algorithm for tor circuit scheduling," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 329–339.

[34] C.-T. Li and M.-S. Hwang, "A lightweight anonymous routing protocol without public key en/decryptions for wireless ad hoc networks," *Information Sciences*, vol. 181, no. 23, pp. 5333–5347, 2011.

[35] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[36] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[37] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 119–132.

[38] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless communications*, vol. 11, pp. 62–67, 2004.

[39] C. J. McIvor, M. McLoone, and J. V. McCanny, "Hardware elliptic curve cryptographic processor over $rmgf(p)$," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 9, pp. 1946–1957, 2006.

[40] P. Palmieri and J. Pouwelse, "Key management for onion routing in a true peer to peer setting," in *International Workshop on Security*. Springer, 2014, pp. 62–71.

[41] B. Poettering, "SECCURE Elliptic Curve Crypto Utility for Reliable Encryption," Online http://point-at-infinity.org/seccure/ (Accessed 28/02/17).

[42] R. Annessi and M. Schmiedecker, "Navigator: Finding faster paths to anonymity," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 214–226.

[43] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks," in *2003 Symposium on Security and Privacy, 2003.*, May 2003, pp. 28–41.

[44] D. Moore and T. Rid, "Cryptopolitik and the darknet," *Survival*, vol. 58, no. 1, pp. 7–38, 2016. [Online]. Available: http://dx.doi.org/10.1080/00396338.2016.1142085

[45] "Some statistics about onions," Online https://blog.torproject.org/blog/some-statistics-about-onions (Accessed 15/02/17), February 2015.