FinTracer: A privacy-preserving mechanism for tracing electronic money

Michael Brand
School of Computing Technologies, RMIT University
Melbourne, Vic, Australia
AUSTRAC
Canberra, ACT, Australia
michael.brand@rmit.edu.au

Hamish Ivey-Law
College of Engineering & Computer Science, The Australian National University
Canberra, ACT, Australia
AUSTRAC
Canberra, ACT, Australia
hamish.ivey-law@anu.edu.au

Tania Churchill
College of Engineering & Computer Science, The Australian National University
Canberra, ACT, Australia
AUSTRAC
Canberra, ACT, Australia
tania.churchill@anu.edu.au

ABSTRACT

Information sharing between financial institutions can uncover complex financial crimes such as money laundering and fraud. However, such information sharing is often not possible due to privacy and commercial considerations, and criminals can exploit this intelligence gap in order to hide their activities by distributing them between institutions, a form of the practice known as "layering".

We describe an algorithm that allows financial intelligence analysts to trace the flow of funds in suspicious transactions across financial institutions, without this impinging on the privacy of uninvolved individuals and without breaching the tipping off offence provisions between financial institutions. The algorithm is lightweight, allowing it to work even at nation-scale, as well as for it to be used as a building-block in the construction of more sophisticated algorithms for the detection of complex crime typologies within the financial data. We prove the algorithm's scalability by timing measurements done over a full-sized deployment.

KEYWORDS

anti money laundering, financial crime, graph analytics, homomorphic encryption, private graph analysis

1 INTRODUCTION

There is no doubt that policing against financial crimes such as money laundering and fraud is easiest when all information regarding a particular crime is pooled together. However, this information is often siloed among the different financial institutions, and cannot be shared due to legal, privacy and commercial considerations.

We introduce *FinTracer*, a privacy-preserving algorithm that provides a solution for the safety-vs-privacy dilemma in the context of financial crime investigation. It allows patterns of criminal activity, known as criminal *typologies*, to be searched for and their occurrences in the data to be retrieved as though the typologies had been queried on a graph database that includes all relevant financial information, but without any breach of privacy.

YYYY(X), 1–12

https://doi.org/XXXXXXX.XXXXXXX

Other algorithms that can be run in a privacy-preserving way across financial institutions exist (e.g., [22]). FinTracer's innovations are that it is both

(1) Extremely lightweight, and
(2) Highly customisable.

The combination of both traits is what allows FinTracer to be used as a powerful building block, and for it to be easily customised to define varied complex typologies, making it suitable in the context of the exceedingly adaptable nature of financial crime.

2 MOTIVATING EXAMPLE

On its own, FinTracer can be used for the detection of fraud; for connecting accounts suspected of laundering money with accounts demonstrating unexplained wealth, for finding connections between persons of interest, or for grouping instances of unrelated-looking suspicious events into larger clusters, just to name a few examples. In conjunction with other privacy-preserving techniques, its uses are even more varied.

For concreteness we begin by introducing one motivating example, which will serve both to explain the constraints within which FinTracer was designed to operate and, later on in the paper, to show how concrete parameter choices for the algorithm can be used to solve different problems in combating financial crime.

Australia's National Disability Insurance Scheme (NDIS) provides eligible Australians, who have a permanent or significant disability, with funding to assist them in their daily lives. The NDIS was introduced "to improve the lives of people with disability and the community more generally by providing insurance for all Australians and lowering future costs of disability support" [1].

According to the National Disability Insurance Agency (NDIA), there are around 4.3 million Australians who have a disability. According to [19], in the years 2020–2025 alone, the NDIS will provide more than AU$22 billion in funding a year, to an estimated 500,000 Australians who have permanent and significant disability.

While such incentives are designed to support people with disabilities, they also provide opportunities for economic crime. According to the Australian Criminal Intelligence Commission (ACIC) chief Michael Phelan, organised criminal gangs have been defrauding billions of dollars from the NDIS scheme [17]. The Australian Government is committed to preventing such fraud to ensure vital funding, such as the NDIS, supports Australians who need it [1].

The *Australian Transaction Reports and Analysis Centre* (AUSTRAC) is Australia's anti-money laundering and counter-terrorism financing regulator and financial intelligence unit. It is the government agency “responsible for preventing, detecting and responding to criminal abuse of the financial system to protect the community from serious and organised crime” [3].

The Anti-Money Laundering and Counter-Terrorism Financing Act 2006 (AML/CTF Act) [13] requires reporting entities to maintain an AML/CTF program. Each reporting entity is required to provide a report to AUSTRAC on International Funds Transfer Instructions, cash deposits, withdrawals equal to or greater than AU$10,000, and cross-border movements, as well as to submit Suspicious Matter Reports (SMRs).

Regulated entities are required to analyse their own data to identify any suspicious activity. If a suspicion is formed that an entity or transaction may be linked to a crime, an SMR must be submitted to AUSTRAC. Reporting entities do not have access to data from other financial institutions in their analysis process. In fact, they “must not disclose any information about an SMR, or do anything which could reasonably infer that [they] have submitted an SMR or are required to submit an SMR about one of [their] customers (except for certain limited circumstances)” [2] – to do so would in most circumstances be a criminal offence.

Data on domestic transactions, i.e. payments made between bank accounts within Australia, are not reportable to AUSTRAC in the absence of a suspicion being formed on an entity.

Criminals may seek to conceal their activities by moving funds across multiple financial institutions. Consider, again, the example of NDIS fraud. One indicator of NDIS fraud may be sending large amounts of NDIS sourced funds overseas [1]. The existence of a single indicator does not necessarily indicate criminal activity, but the presence of multiple indicators may warrant further investigation.

If a criminal fraudulently receives an NDIS payment and immediately moves the funds off-shore, this activity can be detected by the financial institution the criminal banks with. A criminal may, however, seek to conceal their activity by moving the funds domestically across accounts owned by different financial institutions, before ultimately moving the money off-shore. This action (which is a form of *layering*), means that neither the financial institutions that own the accounts, nor AUSTRAC, see the full picture. Without receiving other intelligence, it may not be possible to connect the money moving offshore with the source of the money — in this case from NDIS payments.

On the face of it, detecting this type of crime requires increased data sharing between financial institutions and AUSTRAC.

There is a tension between protecting the privacy of individuals and protecting Australian society through the analysis of personal data in order to identify criminal activity. FinTracer, however, presents an approach that addresses both goals without needing to trade them off.

FinTracer allows for such typologies as the criminal seeking to conceal their NDIS fraud activities by layering to be searched for by AUSTRAC without any breach of privacy for ‘innocent bystanders’, nor any information exchange beyond what is allowed by the AML/CTF Act.

We will show how FinTracer can be used to detect NDIS fraud by appropriate parameter choices, but readers will find it straightforward to adapt the algorithm with other parameter choices to tackle a large variety of investigative scenarios, far beyond this individual example.

3 THE PROBLEM STATEMENT

As with any real-world problem, there is no single, objective problem statement. Many different tools can help intelligence analysts in different ways to pinpoint financial crime. The key to FinTracer was that it needed to be lightweight enough to scale up to the size of the Australian economy,[1], and in addition allow FinTracer to serve as a basic building block in the construction of more advanced algorithms.

To this end, let F be a set of financial institutions, let A be the set of accounts managed by these financial institutions, and let $m : A \rightarrow F$ be a function mapping each account to the financial institution managing it.

Furthermore, for each $f \in F$, let $A^f \overset{\text{def}}{=} \{x \in A | m(x) = f\}$ be the set of accounts managed by f, noting that the A^f partition A.

Additionally, let T be the set of transactions between the accounts in A, and let T^{f+} be the subset of T where money is transferred from an account managed by f and T^{f-} be the subset of T where money is transferred to an account managed by f, noting that both the T^{f+} and the T^{f-} form partitions of T.

The practice in the financial world is that each financial institution, f, maintains substantial amounts of information regarding each account in A^f and each transaction in T^{f+} and T^{f-}. By contrast, a financial institution has no knowledge of any transaction that does not appear in T^{f+} and T^{f-}, and no knowledge of any account that is not in A^f and does not participate in T^{f+} or T^{f-}. AUSTRAC itself has, for our purposes, no knowledge about A and T at all, unless a suspicion exists regarding specific accounts or specific transactions that requires AUSTRAC to be alerted to them.

Our privacy requirements for FinTracer are that it upholds this status quo: FinTracer must not divulge to any financial institution information about any account or any transaction that it is not already privy to, and AUSTRAC must not learn about any account or any transaction that does not have suspicion attached to them. The purpose of FinTracer, however, is to surface new types of suspicion, previously obscured by layering, and to alert AUSTRAC regarding the accounts and transactions involved.

A FinTracer query does not run directly on the rich data set of everything known to the financial institutions. Instead, FinTracer expects its query to relate to a digraph, $G = \langle V, E \rangle$, where E describes those pairs of accounts, (a, b), that exhibit a relation that we want FinTracer to follow, and V is the set of accounts induced by E.

The relation described by $(a, b) \in E$ is not arbitrary. It is a function of the transactions that have occurred between a and b. The transactions in both directions are taken into account, but not in a direction-agnostic manner.

Here is an example of a function, using arbitrary parameter selections, that may generate such an E: An ordered pair (a, b) appears in E if all of the following holds true:

[1]Australia’s Gross Domestic Product (GDP) was $1.423T as of 2020 [11], making it the world’s 13th largest economy. Size metrics directly related to the performance of FinTracer are discussed in Section 8.

- There were no transactions between a and b, in either direction, prior to March 30th 2020,
- There were no transactions from b to a on any date, and
- At least \$10,000 were transferred from a to b since March 30th 2020.

It is assumed that if no transactions occurred in either direction between a and b, then $(a, b) \notin E$.

A FinTracer query is composed of four parts:

G: The *connection digraph*,
S: The *source accounts*, a subset of V,
D: The *destination accounts*, also a subset of V, and
k: The *maximal path length*, a natural.

FinTracer was designed so as to allow an operator on the AUSTRAC node to specify queries. This operator, however, must design their queries without the information of which accounts exist or what transactions occur between them. It therefore cannot actually determine G, S and D. Instead, FinTracer queries are asked in the form $\langle Q_G, Q_S, Q_D, k\rangle$, where each of the first three elements is a function that allows the relevant information to be determined by the receiving financial institutions.

For example, G may be as in the example given above, S may be the set of accounts receiving NDIS payments, and D may be the accounts transferring funds above a certain amount overseas. These descriptions can be sent to the financial institutions (preferably in some unambiguous way, such as by means of a SQL query), and each financial institution can separately resolve them.

In the particular case of the G given in the example, the graph's most salient condition for including (a, b) in $E(G)$ is that a transferred money to b. When this is the case, the algorithm behaves as if it is "following the money". Importantly, however, this is only one type of use. In other cases, we may want to follow the money backwards, and in other cases still (such as in the examples mentioned of attempting to find connections between people or between events) the edges in $E(G)$ are oblivious to the direction of money flow (because such connections are inherently undirected relationships).

Naturally, because no financial institution is privy to the full A or the full T, none of them is able to determine the full G, S or D, however, each f can determine $S^f \stackrel{\text{def}}{=} S \cap A^f$, $D^f \stackrel{\text{def}}{=} D \cap A^f$, and similarly also E^{f+} and E^{f-}, each of which partitions E.

With this preliminary set of definitions, we can now finally define our chosen objective. Namely, given $\langle Q_G, Q_S, Q_D, k\rangle$, we are interested in a privacy-preserving algorithm that returns to AUSTRAC the account set $R = \{y \in D | \exists x \in S, \text{ s.t. there is a path of length at most } k \text{ on } G \text{ from } x \text{ to } y\}$.

Considering, once again, our running example of NDIS fraud, if certain accounts S received NDIS payments and the account owners then moved the funds from account to account across a small number ($\leq k$) of hops until reaching some final accounts, then we are interested in finding those final accounts which transferred money overseas.

To be privacy-preserving, the algorithm must be implemented as a protocol over all involved parties (AUSTRAC and the financial institutions) that ultimately returns R to relevant officers in AUSTRAC, with AUSTRAC learning in this process no new information other than R (and potentially some general, non-privacy-invasive statistics), and with each financial institution f learning in this process no new information other than the parameters of the query and the partial result $R^f = R \cap V^f$.

We note that one may consider alternative problem setups in which f learns even less information. E.g., f may conclude the protocol without learning R^f. However, this algorithm was developed for real use by AUSTRAC and the Australian financial institutions, so was designed to provide tangible benefits to all involved parties (all of whom participated voluntarily). In such a scenario, providing R^f to each f gives each financial institution important information regarding their own risk landscape.

To be of practical use, the chosen algorithm had to be able to scale up to process data from the entire Australian economy, while remaining both frugal in its communication requirements and fast.

4 FINTRACER AS A BUILDING BLOCK

Before going on to detail how FinTracer is implemented, consider its uses in investigating financial crime.

A typology is a pattern of behaviour among one or more entities. Investigators within a financial institution trying to detect occurrences of any particular typology can match the characteristics of entities involved in the typology based on the client information that they collect. Such data is commonly known as *Know Your Customer* (KYC) information. This information must be gathered and retained to establish the legitimacy of a customer's identity and to identify their risk factors [14]. Thus, investigators are able to match profile descriptions against customers, and customers against their accounts. On the other hand, individual institutions typically cannot match the characteristics of *interactions* based on a profile description. This is because such interactions can describe money flows that are not fully visible to any financial institution. This can be for one or both of two reasons:

(1) The flow is indirect and passes through accounts in multiple institutions, and
(2) Part of the profile description relates to attributes of the two accounts at either end, which may not belong to the same institution.

With FinTracer, queries can specify:

- The type of account required on each end of a target flow,
- The type of account required for intermediate accounts,
- The characteristics of the flow itself, and
- The desired level of indirectness for a flow.

On its own, this allows FinTracer to be used in discovering and investigating typologies that involve one connection between two types of accounts of interest, but when combining multiple FinTracer queries, the result is essentially a general graph query, allowing analysts to query for arbitrarily complex typologies.[2]

5 THE BASIC SOLUTION ALGORITHM

We introduce the FinTracer algorithm here in a basic form, adding improvements and extra functionality in later sections.

The FinTracer algorithm works in three phases:

(1) Initialisation,
(2) Propagation (which is repeated k times), and
(3) Result reading.

[2]We discuss methods for combining FinTracer queries in a separate upcoming paper.

We will describe these in turn.

5.1 The FinTracer tag

The core data structure used in the FinTracer algorithm is the *FinTracer tag*. The algorithm's initialisation creates a new tag, propagation manipulates the tag's values, and result reading retrieves information from the tag.

A FinTracer tag is a partial mapping from V to a semi-homomorphically-encrypted value. Specifically, for reasons of execution speed, we use ElGamal encryption [12, 18] over the additive Curve25519 [4] twisted Edwards elliptic curve group [5], storing curve points in their extended projective form [15]. This is the same curve as is used in the Ed25519 digital signature algorithm [7]. It is a form optimised for speed, and is especially suited for GPU implementations, as was done in, e.g., [10]. We developed our own optimised GPU implementation of it based on `libsodium` [6].

In total, this data structure amounts to some subset of the accounts in V being mapped to a single encrypted value, with that value being stored in 256 bytes.

Though the plaintext space for Ed25519 values is substantial, at over 2^{252} elements, additive ElGamal encryption is not conducive to fully decrypting an arbitrary plaintext. As a result, we will only use each plaintext to store a single bit, a zero plaintext will signify "False" and a non-zero plaintext will signify "True". Thus, each account will be associated with a single bit value, but that value will be stored in encrypted form over 256 bytes. (Accounts that are not mapped against any ciphertext will be considered as associated with a "False" value.)

Note that no single participant in the FinTracer algorithm can store a complete tag. This is because the account identifiers that are the domain of the tag's mapping are only known to the financial institutions managing said accounts. To store a tag, we shard it between the participating financial institutions. Each institution f stores the portion of the mapping whose domain is within V^f.

We refer to the Boolean to which a tag maps any particular account as that account's *tag value*.

A crucial point in the FinTracer algorithm is that the key used for the ElGamal encryption is one generated by AUSTRAC. While all participants hold the ElGamal public key, AUSTRAC alone holds the ElGamal private key.

5.2 Initialisation

Different implementation flavours of FinTracer propagation require slightly different initialisations, but the basic principle is always that accounts are mapped to "True" when they are "accounts of interest".

In our specific implementation, we will use two tags, $t_=$ and $t_\leq$. The former will answer the question of which destination accounts are exactly k hops from any source account; the latter — which are *at most* k hops away.

Initially, we will set both $t_=$ and $t_\leq$ to the set of source accounts. Given Q_S, each financial institution can create this tag independently.

In practice, Q_S comes in the form of a database query, typically in SQL, that retrieves those accounts that match the relevant criteria. In our running example, these will be the accounts that received NDIS payments.

Once the accounts are retrieved, a FinTracer tag is created that assigns a "1" to each account in S. Accounts not in S do not need to be assigned values at all.

The full initialisation process is described in Algorithm 1. We assume the existence of a computing system where both AUSTRAC and each of the financial institutions has a separate computation node, allowing for local computation visible only to that party. (In practice, each node may be implemented using one or more computers.) We name the AUSTRAC node "AT", and each node managed by a financial institution is named by its member in F. Here and throughout this document, we use "**on**" blocks to indicate which nodes code runs on. Lines of code that run on multiple nodes can run in parallel. The command "**transmit**" is used to indicate inter-node communication. It sends information from the executing nodes to designated target nodes. The command "**receive**" indicates receipt of the information at the target nodes. It is assumed that all such communication is cryptographically and/or physically protected at the channel level, to ensure that it cannot be intercepted other than by the sending and the receiving party. All other commands run in embarrassingly-parallel fashion.

Because commands run completely independently on the different nodes, we will not, as a general rule, differentiate the variable names based on the nodes they are on. Thus, the variable $t_=$, for example, refers to the local value of this variable on whichever node is currently performing the processing. More rigorously, one can think of these as distinct variables, $t_=^f$. In the code, such superscripts (or subscripts) will only be used where disambiguation is necessary.

Algorithm 1 FinTracer initialisation (Basic version)

```
1:  on AT do:
2:      Generate public/private ElGamal key pair $\langle K_{\text{pub}}, K_{\text{priv}} \rangle$.
3:      transmit $\langle K_{\text{pub}}, Q_G, Q_S, Q_D, k \rangle$ to $F$.
4:  end on
5:  on all $f \in F$ do:
6:      receive $\langle K_{\text{pub}}, Q_G, Q_S, Q_D, k \rangle$ from AT.
7:      $zero \leftarrow Enc_{K_{\text{pub}}}(0)$.
8:      $one \leftarrow Enc_{K_{\text{pub}}}(1)$.
9:      Retrieve $S^f$ based on $Q_S$.
10:     Retrieve $D^f$ based on $Q_D$.
11:     Retrieve $E^{f+}$ and $E^{f-}$ based on $Q_G$.
12:     for all $a \in S^f$ do
13:         $t_=(a) \leftarrow one$.
14:     end for
15:     $t_\leq \leftarrow t_=$.
16: end on
```

5.3 Propagation

The heart of the FinTracer algorithm is the propagation step.

Consider the accounts in A as though each $a \in A$ was represented by a unique vector position $p(a)$. This is impossible to do in practice, because no party is aware of all elements in A, but given such a hypothetical mapping $p(a)$ one can think of a FinTracer tag t as representing an (encrypted) vector $\tilde{t}$, such that $\tilde{t}[p(a)] = t(a)$.

Using the same mapping p, the digraph G can be represented by its adjacency matrix M_G.

The purpose of the propagation step is to compute the matrix by vector product $M_G \tilde{t}$.

The main tool used by the algorithm to allow such propagation efficiently is a property of G we refer to as *two-sided visibility.* This is defined as follows.

Definition 1. A query Q_G defining a graph G is considered *two-sided visible* if the conditions determining for every $a, b \in A$ whether the pair (a, b) is in $E(G)$ are ones that can be resolved independently by both $m(a)$ and $m(b)$.

In particular, we defined Q_G in the context of the FinTracer algorithm as resolving whether $(a, b) \in E(G)$ by use of the set of transactions, in either direction, between a and b. We will require, for the FinTracer algorithm, that the details of these transactions actually used for this determination will be those details visible to both participating parties. This includes all the main data regarding transactions: from party, to party, date, time and amount.

Where Q_G is two-sided visible, Algorithm 2 can be used as a naive way to perform the propagation.

Here, a *refresh* operation on a ciphertext is the summation to it of a never-before-used encryption of zero. This is an operation that we will perform before sending any ciphertext from one node to another, so as to prevent, e.g., repetition attacks.

The propagation process described above updates the tag values as desired. However, it has many redundancies. Algorithm 3 and Algorithm 4 present two methods that can be used to reduce these redundancies, which we refer to as *from-compressing* and *to-compressing*, respectively.

The idea behind both types of compression is that the information carried by P_g^f in its uncompressed form is highly redundant. From-compressing utilises the fact that all $(a, b) \in P_g^f$ with the same a values carry the same encrypted value, and can therefore be transmitted only once. To-compressing utilises the fact that all values associated with $(a, b) \in P_g^f$ with the same b ultimately get summed into the value of $t_=(b)$, so only their sum needs to be communicated.

These compression methods do not reduce the overall time complexity of the algorithm, but reduce the communication volumes from $O(E(G))$ to the typically much smaller $O(V(G))$.

See Section 6 for a discussion of when to use each of the three propagation methods.

Algorithm 2 FinTracer propagation (Uncompressed)

```
on all $f \in F$ do:
    for all $g \in F$ do
        Initialise $P_g^f$ as an empty partial mapping.
        for all $(a, b) \in E^{f+}$ where $m(b) = g$ do
            $P_g^f((a, b)) \leftarrow t_=(a)$.
        end for
        if $g \neq f$ then
            for all $(a, b) \in \mathrm{dom}(P_g^f)$ do
                Refresh $P_g^f((a, b))$.
            end for
        end if
        transmit $P_g^f$ to $g$.
    end for
end on
on all $g \in F$ do:
    Initialise $t_=$ to an empty partial mapping.
    for all $f \in F$ do
        receive $P_g^f$ from $f$.
        for all $(a, b) \in \mathrm{dom}(P_g^f)$ do
            if $b \notin \mathrm{dom}(t_=)$ then
                $t_=(b) \leftarrow P_g^f((a, b))$.
            else
                $t_=(b) \leftarrow t_=(b) + P_g^f((a, b))$.
            end if
        end for
    end for
    for all $b \in \mathrm{dom}(t_=)$ do
        if $b \notin \mathrm{dom}(t_\leq)$ then
            $t_\leq(b) \leftarrow t_=(b)$.
        else
            $t_\leq(b) \leftarrow t_\leq(b) + t_=(b)$.
        end if
    end for
        ▷ We will, in later listings, abbreviate the above loop as "$t_\leq \leftarrow t_\leq + t_=$".
end on
```

5.4 Result reading

After k propagation steps, it is time to read the results. Specifically, the results are stored in $t_\leq$, but they are still in encrypted form and the financial institution holding them is not able to read them. To complete the algorithm, we still need to divulge the following information.

- To AUSTRAC: the identity of those accounts that are in D and have a nonzero value in $t_\leq$;
- To each $f \in F$: the identity of the subset of these accounts that are in D^f.

The financial institution should not receive any additional information, and AUSTRAC should not receive any information beyond general statistics. In particular, AUSTRAC should not be able to determine which accounts are in $\mathrm{dom}(t_\leq)$ or even the size $|\mathrm{dom}(t_\leq)|$. It should also not be able to determine which accounts are in D, unless they have nonzero tag values.

From information-theoretic considerations, it is clear that it is not possible to perform such result reading without AUSTRAC learning some information about the size of each $|D^f|$.[3] Divulging the exact $|D^f|$ is still potentially privacy-invasive, for which reason we will not allow it. However, if the relevant officers in AUSTRAC learn only an approximate size for D^f (which may be commercially sensitive information, but provably not privacy intrusive), this is deemed acceptable.

The result-reading procedure is given in Algorithm 5. It contains the line "Add noise to r^f". Without this line, the algorithm functions, but reveals to AUSTRAC the exact value of $|D^f|$. We will expand

[3]The financial institution needs to communicate to AUSTRAC at least $|D^f|$ bits of information.

Algorithm 3 FinTracer propagation (From-compressed)

```
on all $f \in F$ do:
    for all $g \in F$ do
        Initialise $P^f_g$ as an empty partial mapping.
        for all $a \in V^f$ s.t. $\exists b$, where $(a, b) \in E^{f+}$ and $m(b) = g$ do
            $P^f_g(a) \leftarrow t_=(a)$.
        end for
        if $g \neq f$ then
            for all $a \in \mathrm{dom}(P^f_g)$ do
                Refresh $P^f_g(a)$.
            end for
        end if
        transmit $P^f_g$ to $g$.
    end for
end on
on all $g \in F$ do:
    Initialise $t_=$ to the empty mapping.
    for all $f \in F$ do
        receive $P^f_g$ from $f$.
        for all $(a, b) \in E^{g-}$ where $m(a) = f$ do
            if $b \notin \mathrm{dom}(t_=)$ then
                $t_=(b) \leftarrow P^f_g(a)$.
            else
                $t_=(b) \leftarrow t_=(b) + P^f_g(a)$.
            end if
        end for
    end for
    $t_\leq \leftarrow t_\leq + t_=$.
end on
```

Algorithm 4 FinTracer propagation (To-compressed)

```
on all $f \in F$ do:
    for all $g \in F$ do
        Initialise $P^f_g$ as an empty partial mapping.
        for all $b$ s.t. $\exists a$, where $(a, b) \in E^{f+}$ and $m(b) = g$ do
            if $b \notin \mathrm{dom}(P^f_g)$ then
                $P^f_g(b) \leftarrow t_=(a)$.
            else
                $P^f_g(b) \leftarrow P^f_g(b) + t_=(a)$.
            end if
        end for
        if $g \neq f$ then
            for all $b \in \mathrm{dom}(P^f_g)$ do
                Refresh $P^f_g(b)$.
            end for
        end if
        transmit $P^f_g$ to $g$.
    end for
end on
on all $g \in F$ do:
    Initialise $t_=$ to the empty mapping.
    for all $f \in F$ do
        receive $P^f_g$ from $f$.
        for all $b \in \mathrm{dom}(P^f_g)$ do
            if $b \notin \mathrm{dom}(t_=)$ then
                $t_=(b) \leftarrow P^f_g(b)$.
            else
                $t_=(b) \leftarrow t_=(b) + P^f_g(b)$.
            end if
        end for
    end for
    $t_\leq \leftarrow t_\leq + t_=$.
end on
```

later on, in Algorithm 6, how noise is added, to hide the set's exact size.

Some notes regarding Algorithm 5:

(1) The reason we add zero-valued entries to r^f is to ensure that the starting size of r^f is exactly $|D^f|$. We ultimately do not want AUSTRAC to learn the exact size of D^f, and our algorithm adds noise to make sure that this is not revealed. However, AUSTRAC nevertheless learns the approximate value of $|D^f|$. Based on the specifics of the real-world problem, we have determined that this particular information leak of non-private general statistics is allowable. However, if r^f is not padded to $|D^f|$ size, AUSTRAC will learn the approximate value of another statistic, for which we cannot determine whether it is an allowed information leak or not.

(2) "Sanitising" a ciphertext refers to multiplying it by a random, uniformly-chosen, non-zero integer in the cyclic group modulo the size of the elliptic curve. The result is zero if the original value was zero, and otherwise is a uniformly distributed non-zero value. As a result, this operation erases all information in a tag value other than its zero/non-zero status. When sanitising an entire tag, each tag value is sanitised using an independent random multiplier.

(3) It is not possible to fully decrypt an element in our chosen encryption scheme. The result of decryption is a Curve25519 group element, not the plaintext from which it was generated. Returning to the plaintext requires a discrete logarithm, which is assumed to be a hard problem as part of the security assumptions of the ElGamal cryptosystem. However, while it is not possible to decrypt the value, it is nevertheless possible to compare two Curve25519 elements for equality. This is how the condition "$Dec_{K_{\mathrm{priv}}}(v^f[i]) \neq 0$" is evaluated.

To complete the description, we still need to explain line 7 of Algorithm 5.

The main idea is that in order to hide the exact size of D^f, we add to the mapping r^f a randomly chosen number x^f of "fake" entries. These are entries whose domain is not an account at all, but they cannot be selected by AUSTRAC because their values are all (encrypted) zeroes.

To determine how to best choose x^f, it is important to understand why this noise is needed. If an adversary is purely interested in knowing the size of D^f (e.g., for competitive business intelligence) then knowing its approximate size is likely to be equally effective. The reason to hide the exact size is therefore different. Namely, it is purely for the purpose of privacy preservation. We

Algorithm 5 Result reading (Basic version)

```
1:  on all $f \in F$ do:
2:      Let $r^f$ be $t_{\leq}$ reduced to only its entries in $D^f$.
3:      for all $a \in D^f \setminus \mathrm{dom}(r^f)$ do
4:          $r^f(a) \leftarrow zero$.                    ▷ See Note 1, above.
5:      end for
6:      Sanitise $r^f$.                              ▷ See Note 2, above.
7:      Add noise to $r^f$.                          ▷ See Algorithm 6.
8:      Let $\pi$ be a uniformly-chosen random bijection from
    $1, \dots, |r^f|$ to the domain of $r^f$.
9:      for all $i \in 1, \dots, |r^f|$ do
10:         $v^f[i] \leftarrow r^f(\pi(i))$.
11:     end for
12:     transmit $v^f$ to AT.
13: end on
14: on AT do:
15:     for all $f \in F$ do
16:         receive $v^f$ from $f$.
17:         for all $i \in 1, \dots, |v^f|$ do
18:             $d^f[i] \leftarrow Dec_{K_{\mathrm{priv}}}(v^f[i]) \neq 0$.   ▷ See Note 3, above.
19:         end for
20:         transmit $d^f$ to $f$.
21:     end for
22: end on
23: on all $f \in F$ do:
24:     receive $d^f$ from AT.
25:     $R^f \leftarrow \{\}$.
26:     for all $i \in 1, \dots, |d^f|$ do
27:         if $d^f[i]$ then
28:             $R^f \leftarrow R^f \cup \{\pi(i)\}$.
29:         end if
30:     end for
31:     transmit $R^f$ to AT.
32: end on
33: on AT do:
34:     $R \leftarrow \{\}$.
35:     for all $f \in F$ do
36:         receive $R^f$ from $f$.
37:         $R \leftarrow R \cup R^f$.
38:     end for
39: end on
40:                         ▷ $R$ is the final result retrieved by AUSTRAC.
41:                              ▷ $R^f$ is the partial result learned by $f$.
```

wish to avoid revealing the information of whether particular accounts appear in D^f (other than those directly retrieved by the query), even in a scenario where AUSTRAC is free to craft D^f so that it either includes or does not a specific account of interest, or where AUSTRAC is free to run multiple queries that differ only by a single account of interest.

To this aim, we choose x^f so as to satisfy the criterion of (ϵ, δ)-differential privacy. A probabilistic algorithm, $\mathcal{A}$ is said to be (ϵ, δ)-*differentially private* if for any input X (described as a set), with probability at least $1 - \delta$, the output of $\mathcal{A}$ on X, v, is such that for any input Y that is different to X by only a single element,

$$e^{-\epsilon}\mathrm{Prob}[\mathcal{A}(Y) = v] \leq \mathrm{Prob}[\mathcal{A}(X) = v] \leq e^{\epsilon}\mathrm{Prob}[\mathcal{A}(Y) = v].$$

The idea behind (ϵ, δ)-differential privacy is that it is "almost always" the case that it is "almost impossible" to tell whether any element was added to or subtracted from the set of interest, which, in our case, is the set of accounts $D^f \setminus R^f$. This provides, for every account (except for those in R^f), protection against leakage of the information of whether or not they are in $D^f \setminus R^f$, thus protecting their privacy.

The (ϵ, δ)-differential privacy paradigm differs from its more popular cousin, ϵ-differential privacy, in that ϵ-differential privacy guarantees that it is *always* the case that it is "almost impossible" to tell whether any element was added to or subtracted from the set of interest. In our case, this would not have been possible, however, because when hiding a value in ϵ-differential privacy by adding to it a noise factor Δ, that Δ must have a positive probability of being any integer. In particular, it must have a positive probability of being negative. In our case, we can only add fake entries to r^f, not subtract them, so our x^f must be nonnegative.

If D^f is of some size $|D^f|$ and x^f happens to be chosen to be zero, AUSTRAC will be able to tell that the true value of $|D^f|$ cannot be one more than this. As a result, at least at $x^f = 0$, we cannot satisfy ϵ-differential privacy. We can, however, require the next best thing, which we refer to as *strict* (ϵ, δ)*-differential privacy.*

Definition 2. An algorithm that adds a non-negative, integer number, x, of fake entries into a set in order to provide (ϵ, δ)-differential privacy will be said to satisfy *strict* (ϵ, δ)*-differential privacy* if the probability that $x = 0$ is at most δ, and whenever $x > 0$, the conditions of ϵ-differential privacy are satisfied.

Theorem 1. *Algorithm 6 describes an algorithm for determining x, the number of fake entries to add to a set, such that*

(1) *Adding x fake entries to the result set provides strict (ϵ, δ)-differential privacy to an algorithm reporting the size of the result set after the addition of the fake entries, and*
(2) *Algorithm 6 is optimal, in the sense that this distribution for x has the minimal expectation of any distribution satisfying strict (ϵ, δ)-differential privacy.*

Algorithm 6 Adding (ϵ, δ)-differential privacy

```
1:  on all $f \in F$ do:
2:      $\gamma \leftarrow 1 - e^{-\epsilon}$.
3:      $Y \leftarrow \max\left(0, \left\lceil \log\left(\frac{\gamma(\gamma-\delta)}{\delta(1-e^{-2\epsilon})} + 1\right) / \epsilon \right\rceil\right)$.
4:      $t \leftarrow 1 + (\delta - 1)e^{-\epsilon} - \delta e^{(Y-1)\epsilon}$.
5:      $r \leftarrow \mathrm{RandomFloat}(1 - \gamma/t, 1)$.   ▷ Uniform between these
    bounds. Independent between $f$s.
6:      if $r > 0$ then
7:          $x^f \leftarrow Y + \lfloor -\log(r)/\epsilon \rfloor$.
8:      else
9:          $x^f \leftarrow Y + \left\lfloor \log\left(1 + \frac{rt}{\delta e^{(Y-1)\epsilon}}\right) / \epsilon \right\rfloor$.
10:     end if
11:     Add $x^f$ fake accounts with encrypted zero values to $r^f$.
12: end on
```

Proof. consider any distribution P over the nonnegative integers such that for every $\boldsymbol{x} \sim P$, $\mathbf{Prob}[\boldsymbol{x} = x] \leq e^{\epsilon}\mathbf{Prob}[\boldsymbol{x} = x + 1]$. Such a distribution is necessarily a weighted average of shifted exponential distributions, E_y, where the probability for $\boldsymbol{x} \sim E_y$ to be any value $x \geq y$ is proportional to $\exp(-\epsilon(x - y))$ and zero otherwise.

Consider any distribution created by such a weighted average that also meets all other criteria for strict (ϵ, δ)-differential privacy. The weight α_0 associated with E_0 cannot be greater than what would give $\mathbf{Prob}[\boldsymbol{x} = 0] = \delta$, and for all $y > 0$ the weight α_y associated with any E_y cannot be greater than what would give for $\boldsymbol{x_y} \sim E_y$

$$\mathbf{Prob}[\boldsymbol{x_y} = y] = \delta \exp(\epsilon(y - 1))(\exp(\epsilon) - \exp(-\epsilon)).$$

This is because $\delta \exp(\epsilon(y - 1))$ is the greatest possible value for $\mathbf{Prob}[\boldsymbol{x} = y - 1]$ and $\exp(\epsilon) - \exp(-\epsilon)$ is the greatest possible value for $\alpha_y / \mathbf{Prob}[\boldsymbol{x} = y - 1]$.

If there exists a strictly (ϵ, δ)-differential-privacy-preserving distribution D described by a weighted average α_y over all E_y for which it is true that there exists a Y such that for all $y < Y$, α_y attains its maximum possible value, and for all $y > Y$, α_y equals zero, by necessity this distribution is the unique global minimiser of the expectation among all such distributions.

The distribution described by Algorithm 6 meets these criteria, for which reason it is the unique optimum. The full technical calculations that demonstrate this are in Appendix A. □

6 ENHANCEMENTS

Throughout, we make the point that the algorithms presented are naive implementations of the FinTracer idea. Below are some methods that we have used to improve the efficiency of the algorithm in a practical setting.

- **Nonce stockpiles:** One time-consuming part of the algorithm is the refresh needed for every communicated tag value at each propagation round. Refreshing involves encrypting a zero value, then adding that zero value to the existing tag value. While the addition is fast, encryption is much slower. Fortunately, it is also completely independent of the problem parameters. It is therefore possible to generate a stockpile of encrypted zeroes offline, before the algorithm is invoked, in order to speed up its online performance.
- **Compressed sending format:** As described above, what is sent between computation nodes in each propagation round is a mapping (either from accounts or from account pairs) to encrypted values. In practice, this not only carries much overhead, but also runs the risk that the data structure holding the mapping will include unintended information that will in this way be leaked to the recipient. A better implementation is one where each pair (f, g) of financial institutions sends in each round only a vector of tag values, without any additional structure. This can be done by each such (f, g) coordinating at initialisation time the length of the vector and the identity of the account / account-pair associated with each vector position. Because the propagation condition is two-sided visible, each node can separately compute the length necessary for the vector and the set that needs to be mapped to its positions, so such coordination can occur without any information being leaked. The actual ordering of the values can be chosen randomly, to ensure that it contains no exploitable information.
- **One-sided criteria:** In a real-world implementation, many criteria of intelligence value are not two-sided criteria. Every financial institution knows a great deal about the owners of each account, so is in a good position to determine whether certain accounts or certain graph connections are innocuous and can be ignored. Ignoring a signal can be done by the relevant financial institution substituting the corresponding tag value with an encrypted zero before processing it further. However, note that depending on the nature of the filtering involved, such filtering may prohibit the use of particular compression schemes. This is why all three compression schemes (uncompressed, from-compressed and to-compressed) are needed.
- **Size limits:** Our implementation depicts the FinTracer result to always be reported to AUSTRAC. In practice, if the result set is too large, this indicates that the query was from the get-go not a privacy-preserving query. In a real implementation, both AUSTRAC and the financial institutions should first check what size their result sets are, and abort further execution unless these are smaller than a given threshold.
- **Noise on result set size:** In our implementation, we only added differential noise to the number of non-matching accounts that are tracked. There was no need to obfuscate the number of matching accounts, as this account set was going to be reported to AUSTRAC in any case. However, if the process may be aborted, as per the previous point, it is important to also add noise in the form of fake matches, not just fake non-matches. Algorithm 6 can be used to determine the number of the fake matches, too.
- **Commitment:** In a scenario where it is possible for AUSTRAC to report to the financial institutions which accounts are a match, but for the institutions to then reply with an account set that is smaller than what AUSTRAC expected (due to the added noise), it is important for the financial institutions to first make a cryptographic commitment to AUSTRAC regarding the number of fake matches. Otherwise, a financial institution could use the protocol in order to gain information from AUSTRAC regarding the value of particular tags.[4]

7 CORRECTNESS

Theorem 2. *The FinTracer algorithm returns the subset of the destination set D composed of the accounts a to which there is a path of length up to k on G from any element in S, unless the number of walks on G of length up to k from S to a is a multiple of the Curve25519 group size.*

Proof. By induction, the value, under the encryption, of $t_{\leq}(a)$ and $t_{=}(a)$ for any account a after i propagation rounds of the algorithm is precisely the number of walks of length up to i and of length exactly i, respectively, from S to a, modulo the group size.

[4]It is presumed that once AUSTRAC receives a result set, it will continue investigating it in the clear, so will be able to confirm that all results it received are true matches. Together with the commitment regarding the number of fake matches, this does not leave any room for a player to plant in the set of tag values to be decrypted by AUSTRAC any whose value does not correspond to what the protocol expects them to be.

Because the shortest such walk is also a path, the number of walks counted by $t_{\leq}(a)$ is nonzero if and only if the number of paths is nonzero. If the number of walks is not a multiple of the group size, then this corresponds to whether $t_{\leq}(a)$ is an encrypted nonzero value or an encrypted zero value. The value ultimately communicated to AUSTRAC, based on which the result set is determined, is $t_{\leq}(a)$ after k propagation rounds, multiplied by a uniform nonzero group element. This multiplication preserves whether a value is zero or not.[5] □

We deem the case where the number of walks is precisely a multiple of the group size unrealistic, both on practical grounds and because the group size is large enough for the number of walks to never reach it when the algorithm is invoked with practically-beneficial choices for the algorithm parameters, for which reason in normal conditions this caveat can be safely ignored.

8 PERFORMANCE

We tested the behaviour of the system against real Australian transaction data, kindly provided by AUSTRAC, in order to validate that the system is able to detect actual crime typologies and actual instances of financial crime in the real dataset. However, for the purpose of measuring the system's ability to scale out (as well as to provide results that will be reproducible without access to classified data), we ran the system on synthetic, random graphs scaling up to the size of the full Australian monthly transaction graph. These graphs were generated using the R-MAT method [9] with parameters $(0.57, 0.19, 0.19, 0.05)$, following, e.g., [8, 16].

In terms of complexity, the algorithm, in either its from-compressed or to-compressed variants, requires for k FinTracer propagation steps $k|E|$ ciphertext summations, with the complexity of all other operations being at most $O(k|V|)$. The complexity of communication size required for the same operation is $O(k|V|)$ and the complexity of required memory size is $O(|V|)$.

According to [20], the number of transactions in Australia in a typical month is roughly 2^{28}. This is an upper bound on $|E|$, because many of these transactions will represent the same edge in G. (The actual transaction graph is a multigraph, not an ordinary digraph.)

The number of customers served by each of Australia's four major financial institutions is approximately 15 million [21, 23], or roughly 2^{26} customers across all Australian banks. We therefore estimate $|V|$ at 2^{27}, for an average of 2 accounts per customer.

We generated, using R-MAT, random graphs with this $|V|$ and $|E|$. To show the system's scale-out behaviour, a subset of the vertices was chosen, and the algorithm was re-ran on the sub-graph induced by these vertices. Each such experiment was ran 5 times, to demonstrate the system's consistent performance in each case.

Tables 1, 2 and 3 list the timing results for the initial FinTracer propagation step (from a known-sized set of source accounts), for a subsequent FinTracer propagation step (at this point managing $|V|$ tags) and for final result reading, respectively. The graphs depict the view of a single processing node, corresponding to a single f. Timing information is listed against the size of $E^f = E^{f+} \cup E^{f-}$, the set of edges of the propagation graph visible to f, and the size of S^f, the set of source accounts managed by f.

[5]In some scenarios, a user may actually wish to test for the number of walks of a given length k exactly, rather than walks of length up to k. When this is the case, it is possible to read $t_{=}$ instead of $t_{\leq}$.

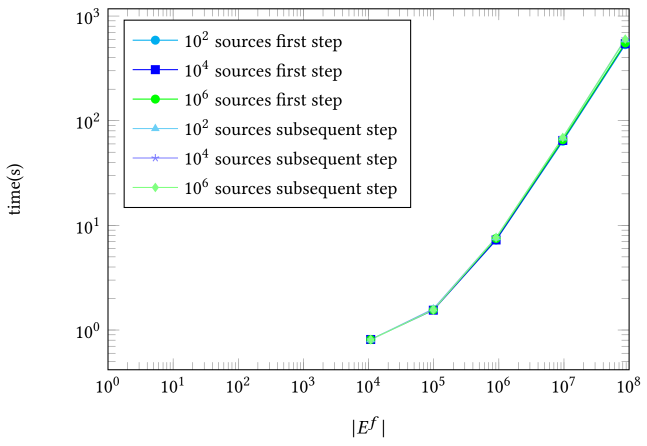

Figure 1: Propagation time scales essentially linearly with $|E^f|$, and is largely independent of all other factors (for which reason all six plots overlap almost completely).

Run times were measured on a distributed deployment on the AWS cloud. Each of Australia's four major banks was represented in the experiment by a single EC2 `g4dn.12xlarge` node. These nodes were chosen due to their large memory sizes (192GB each). Despite the algorithm being straightforward to parallelise, code only used one vCPU out of the 48 available on each EC2 instance, and only one NVIDIA T4 GPU (each having 16GBs of memory) of the 4 available. Thus, these timing measurements refer to non-parallelised, single CPU/GPU runs.

Altogether, the following parameters were tested:

- Number of directed edges, from approximately 10,000 to approximately 100 million going through each node.
- Number of source accounts, from 100 to 1,000,000.
- Number of overall accounts, from approximately 500,000 to approximately 50 million in each node.

These were tested across

- Multiple iterative propagation steps.
- Five tests per configuration.
- Four transaction-holding nodes, simulating Australia's four major banks.

The results show that the system scales linearly with the number of transactions, that even at full scale it completes a propagation step in roughly 600 CPU+GPU seconds, and that these numbers do not change based on how many accounts have positive values or what the original tag size was. This is demonstrated also in Figure 1.

In all experiments, the number of target accounts to be read was kept at 100, this simulating the realistic scenario where not too many results can be returned, in order to maintain privacy. The table shows that reading time is independent of the size of the propagation graph, as well as of the number of sources tagged.

Regarding communication sizes, in our implementation ciphertexts are transported between nodes in their extended projective representation. At 256 bytes per ciphertext, this translates to a total of 32GB communicated each round, for $|V| = 2^{27}$. This could have

Table 1: FinTracer first-step propagation times (seconds).

$\lvert E^f \rvert$	$\lvert S^f \rvert$	min	max	mean	std	relative std
10775	100	0.8	0.8	0.8	0.02	2.1%
10775	10000	0.8	0.8	0.8	0.02	3.0%
10775	1000000	0.8	0.8	0.8	0.03	3.4%
98711	100	1.5	1.6	1.6	0.04	2.9%
98711	10000	1.5	1.6	1.5	0.06	3.7%
98711	1000000	1.5	1.7	1.6	0.08	4.8%
907565	100	7.0	7.4	7.2	0.17	2.4%
907565	10000	7.1	7.5	7.3	0.18	2.5%
907565	1000000	7.3	7.8	7.5	0.17	2.3%
9586273	100	62.7	64.8	63.6	0.70	1.1%
9586273	10000	63.9	65.8	64.8	0.65	1.0%
9586273	1000000	65.4	66.2	65.8	0.32	0.5%
87288800	100	523.7	540.2	533.0	5.50	1.0%
87288800	10000	534.6	553.1	544.2	6.91	1.3%
87288800	1000000	541.5	564.5	552.6	7.52	1.4%

Table 2: FinTracer subsequent-step propagation times (seconds).

$\lvert E^f \rvert$	$\lvert S^f \rvert$	min	max	mean	std	relative std
10775	100	0.8	0.8	0.8	0.02	2.5%
10775	10000	0.8	0.8	0.8	0.03	3.4%
10775	1000000	0.8	0.8	0.8	0.03	3.5%
98711	100	1.5	1.6	1.6	0.04	2.6%
98711	10000	1.5	1.6	1.6	0.04	2.8%
98711	1000000	1.5	1.6	1.6	0.05	3.4%
907565	100	7.4	7.9	7.7	0.16	2.0%
907565	10000	7.4	7.9	7.6	0.16	2.1%
907565	1000000	7.4	7.9	7.6	0.20	2.6%
9586273	100	67.8	69.3	68.6	0.56	0.8%
9586273	10000	68.3	69.7	69.0	0.51	0.7%
9586273	1000000	68.5	70.3	69.1	0.62	0.9%
87288800	100	587.2	599.2	593.2	4.49	0.8%
87288800	10000	597.1	614.7	605.2	5.72	0.9%
87288800	1000000	597.6	610.9	601.6	5.21	0.9%

easily been compressible by a factor of 4 by converting the ciphertexts to a more compressed representation before communicating them, at negligible addition to the computation times.

We see, therefore, that in both communication amounts and computation speeds, even at the scale of a national economy our system is not only practical, but also allows for interactive work, and for FinTracer to be used as a building block for other, more complex algorithms.

9 PRIVACY PRESERVATION

Theorem 3. *If all parties follow the FinTracer algorithm protocol, the financial institutions gain from the data sent to them no knowledge other than the algorithm's inputs $\langle Q_G, Q_S, Q_D, k\rangle$ and the algorithm's output (each f learns the subset of D^f matching the query). AUSTRAC gains from the data sent to it no knowledge other than the approximate size of each D^f, and the algorithm's outputs (the subset of D that matches the query).*

Proof. During the propagation steps of the protocol, all parties exchange between them only encrypted values in an amount that is known ahead of time to the receiving parties. The encrypted values are all refreshed, and therefore unique, and the receiving party does not have the decryption key. Thus, propagation does not add knowledge to any party.

In the result reading step, AUSTRAC receives tags in an amount the provides the approximate value of $|D^f|$, but these tags have been sanitised, and therefore carry only the information of whether they are zero or nonzero. Furthermore, their order has been permuted with a random permutation, so, in fact, only the total number of nonzero entries is revealed to AUSTRAC. This total is, however, the size of the result, so is not additional information given to AUSTRAC. □

For completion, we note that in terms of non-data attacks on this algorithm, the only additional information that can be divulged by

Table 3: FinTracer reading times (seconds).

$\lvert E^f \rvert$	$\lvert S^f \rvert$	min	max	mean	std	relative std
10775	100	0.09	0.11	0.10	0.007	6.8%
10775	10000	0.10	0.13	0.11	0.011	10.4%
10775	1000000	0.10	0.11	0.11	0.006	5.9%
98711	100	0.09	0.12	0.10	0.010	9.3%
98711	10000	0.09	0.11	0.10	0.004	4.2%
98711	1000000	0.09	0.11	0.10	0.005	4.8%
907565	100	0.09	0.11	0.10	0.006	5.9%
907565	10000	0.10	0.11	0.10	0.004	4.0%
907565	1000000	0.10	0.12	0.11	0.008	7.9%
9586273	100	0.09	0.11	0.10	0.006	5.6%
9586273	10000	0.09	0.11	0.10	0.006	6.2%
9586273	1000000	0.09	0.12	0.11	0.011	10.2%
87288800	100	0.09	0.12	0.11	0.010	8.9%
87288800	10000	0.10	0.11	0.11	0.002	2.4%
87288800	1000000	0.09	0.12	0.11	0.011	10.4%

the parties are general bounds regarding the size of E^{f+} by virtue of the time it takes each financial institution to process the data.[6] However, financial institutions can mask this by performing extra summations so as to always round the number of summations up to some upper bound such as 2^{27}.[7]

In a typical analysis of the privacy preservation properties of computing protocols, one would now consider also the possibility that some computing parties may maliciously diverge from the protocol and/or collude. In the case of this system these eventualities are less of a concern because AUSTRAC is a governmental agency and the financial institutions are regulated by AUSTRAC. The FinTracer protocol, accordingly, knowingly does not protect against some extreme eventualities, the most notable of which being a collusion between AUSTRAC and a financial institution in order to gain (some) information regarding financial transactions in other institutions. It also does not protect against AUSTRAC receiving incorrect information if participating financial institutions decide to report untruths. The main protection that it does offer is the following.

Theorem 4. *If financial institutions collude and/or are untruthful, this does not breach the privacy guarantees of Theorem 3 for the owners of accounts in non-colluding financial institutions.*

Proof. The proof of Theorem 3 continues to apply, as nowhere did it rely on the truthfulness of the inputs provided to any party, or any other semantic property of it, other than the structure of the input. □

ACKNOWLEDGMENTS

This work was funded by AUSTRAC.

REFERENCES

[1] AUSTRAC. 2020. NATIONAL DISABILITY INSURANCE SCHEME FRAUD PREVENTION FINANCIAL CRIME GUIDE. https://www.austrac.gov.au/sites/default/files/2020-12/NDIS%20Fraud%20Prevention%20Financial%20Crime%20Guide.pdf.

[2] AUSTRAC. 2021. REGULATORY QUICK GUIDE: Tipping Off. https://www.austrac.gov.au/sites/default/files/2021-06/Quick%20guide%20-%20Tipping%20off.pdf.

[3] AUSTRAC. 2022. AUSTRAC Overview. https://www.austrac.gov.au/about-us/austrac-overview.

[4] Daniel J Bernstein. 2006. Curve25519: new Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography*. Springer, 207–228.

[5] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. 2008. Twisted Edwards curves. In *International Conference on Cryptology in Africa*. Springer, 389–405.

[6] Daniel J Bernstein and Frank Denis. 2019. Libsodium – A modern, portable, easy to use crypto library. https://github.com/jedisct1/libsodium. Retrieved 5 July 2022.

[7] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2011. High-speed high-security signatures. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 124–142.

[8] Harald Bögeholz, Michael Brand, and Radu-Alexandru Todor. 2020. In-database connected component analysis. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1525–1536.

[9] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 2004. R-MAT: A recursive model for graph mining. In *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 442–446.

[10] Shujie Cui, Johann Großschädl, Zhe Liu, and Qiuliang Xu. 2014. High-speed elliptic curve cryptography on the NVIDIA GT200 graphics processing unit. In *International Conference on Information Security Practice and Experience*. Springer, 202–216.

[11] United Nations Statistics Division. 2021. United Nations: National Accounts Main Aggregates Database. https://unstats.un.org/UNSD/snaama/Index. Data upload: December 2021.

[12] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[13] Australian Government. 2021. Anti-Money Laundering and Counter-Terrorism Financing Act 2006. https://www.legislation.gov.au/Details/C2021C00243. Registered 30 June 2021.

[14] Australian Government. 2022. Anti-Money Laundering and Counter-Terrorism Financing Rules Instrument 2007. https://www.legislation.gov.au/Series/F2007L01000. Registered 21 July 2022.

[15] Huseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. 2008. Twisted Edwards curves revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 326–343.

[16] Raimondas Kiveris, Silvio Lattanzi, Vahab Mirrokni, Vibhor Rastogi, and Sergei Vassilvitskii. 2014. Connected Components in MapReduce and Beyond. In *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 1–13.

[17] Nick McKenzie and Amelia Ballinger. 2022. 'The scumbag scale': How organised crime has infiltrated the NDIS. https://www.smh.com.au/national/the-scumbag-scale-how-organised-crime-has-infiltrated-the-ndis-20220811-p5b95s.html. The Sydney Morning Herald.

[6] For simplicity, we assume here that $E^{f+} \gg V^f$.

[7] Similar upper-bounding can also be used instead of the addition of differential noise to hide the size of D^f in the result reading stage, if desired.

[18] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. 2018. *Chapter 8.4 ElGamal public key encryption.* CRC press, 283–319.

[19] NDIS. 2021. What is the NDIS? https://www.ndis.gov.au/understanding/what-ndis.

[20] Reserve Bank of Australia. 2022. Payments data. https://www.rba.gov.au/payments-and-infrastructure/resources/payments-data.html. C6.1 Direct entry and NPP – Original series.

[21] House of Representatives. 2019. Australia's four major banks and other financial institutions: Four major banks. https://parlinfo.aph.gov.au/parlInfo/download/committees/commrep/86eef00b-f8cd-4c13-b8c3-3cef597b3142/toc_pdf/Standing%20Committee%20on%20Economics_2019_11_08_7339_Official.pdf;fileType=application%2Fpdf#search=%22committees/commrep/86eef00b-f8cd-4c13-b8c3-3cef597b3142/0000%22. In *Standing Committee on Economics*, Nov 8 2019.

[22] Alex Sangers, Maran van Heesch, Thomas Attema, Thijs Veugen, Mark Wiggerman, Jan Veldsink, Oscar Bloemen, and Daniël Worm. 2019. Secure multiparty PageRank algorithm for collaborative fraud detection. In *International Conference on Financial Cryptography and Data Security.* Springer, 605–623.

[23] Statista. 2022. Number of customers of major banks Australia 2020. https://www.statista.com/statistics/1211679/australia-big-four-banks-number-of-customers/. Published 2 Apr 2022.

A DIFFERENTIAL PRIVACY CALCULATIONS

We present here the calculations proving that Algorithm 6 yields a value x^f that is distributed according to the distribution that is the unique solution described in the proof to Theorem 1.

Let $\boldsymbol{x}$ be a random value distributed according to the desired distribution. The distribution has the following characteristics, which fully define it.

(1) The support of the distribution is $\mathbb{Z}^{\geq 0}$.
(2) $\mathbf{P}(\boldsymbol{x} = 0) \leq \delta$.
(3) There is some Y such that for all $0 \leq y < Y$, $\mathbf{P}(\boldsymbol{x} = y) = \mathbf{P}(\boldsymbol{x} = 0)e^{\epsilon y}$.
(4) For said Y, for all $y > Y$, $\mathbf{P}(\boldsymbol{x} = y) = \mathbf{P}(\boldsymbol{x} = Y)e^{-\epsilon(y-Y)}$.
(5) For all y, $\mathbf{P}(\boldsymbol{x} = y) \leq e^{\epsilon}\mathbf{P}(\boldsymbol{x} = y+1)$.
(6) For all $y > 0$, $\mathbf{P}(\boldsymbol{x} = y) \leq e^{\epsilon}\mathbf{P}(\boldsymbol{x} = y-1)$.
(7) If $Y > 0$, $\mathbf{P}(\boldsymbol{x} = 0) = \delta$.

To show that the distribution generated by Algorithm 6 meets all criteria when using the Y value derived in the algorithm itself, consider first that because it returns an x^f value in every case, what the algorithm generates is certainly a distribution, in the sense that if we sum up the probabilities for all possible values of x^f we will necessarily reach 1.

Algorithm 6 relies on generating a uniform r in the range $[1 - \gamma/t, 1]$. Importantly, if we compute $\gamma - t$ we get

$$\gamma - t = \delta e^{-\epsilon}\left(e^{Y\epsilon} - 1\right). \qquad (1)$$

By its construction in Algorithm 6 we know $Y \geq 0$, so from (1) we get that $\gamma \geq t$ and therefore $1 - \gamma/t \leq 0$.

We can therefore divide the range of r to $[1 - \gamma/t, 0)$ and $(0, 1]$. (ignoring zero-probability events like $r = 0$). Specifically, we will show that if r is in the range $[1 - \gamma/t, 0)$, this contributes to the part of the distribution with y values lower than the critical Y, and if r is in $(0, 1]$, the remainder.

Consider first the case $r \in (0, 1]$. In this case, the value of x^f is set to $Y + \lfloor -\log(r)/\epsilon \rfloor$. Thus, at $r = 1$, x^f receives the value Y, and the lower r is the greater x^f will be, until as r approaches 0, x^f approaches infinity. The key is that the critical values for r in which the value of x^f changes have a spacing between them that decreases each time by a factor of $e^{-\epsilon}$, so as long as r is uniformly distributed in this range, which it is, the resulting distribution will satisfy criterion 4.

For exactly the same reason, when $r \in [1 - \gamma/t, 0)$ and

$$x^f = Y + \left\lfloor \log\left(1 + \frac{rt}{\delta e^{(Y-1)\epsilon}}\right)/\epsilon \right\rfloor, \qquad (2)$$

we get probabilities that rise in accordance with criterion 3. The fact that a linear transformation is applied on r prior to the log does not change this, because the distribution of the result after the linear transformation is still uniform within a given range.

To verify the edge conditions we plug $r = 0$ and $r = 1 - \gamma/t$ into (2). The former yields $x^f = Y$ directly. For the latter, we can use (1) to determine that it yields the threshold point for $x^f = 0$. Both fall on critical values for r.

By construction, any x^f output by the algorithm is an integer, so combined with the above this also proves criterion 1.

To satisfy criteria 2 and 7, that the probability of $x^f = 0$ is bounded by δ and equals it exactly if $Y > 0$, we calculate the critical value for r that delineates those r values that yield $x^f = 0$ from those that yield $x^f = 1$ in (2). This turns out to be $\delta\gamma/t$, but because the distribution for r was uniform over a range of length γ/t, this translates to a probability of δ exactly.

The other alternative to get $x^f = 0$ is from positive r values, in the case $Y = 0$. However, as can be seen in the definition of Y on Line 3 of Algorithm 6, Y can only be zero if $\gamma \leq \delta$, meaning that $1 - e^{-\epsilon}$, the probability for x^f to be zero (noting that r in this case ranges in $(0, 1]$) is at most δ, as desired.

Lastly, we want to show that the formula for Y in Line 3 of Algorithm 6 is such that conditions 5 and 6 are both satisfied.

For this, consider the values of α_t used in the proof of Theorem 1. These can be computed as

$$\alpha_y = \mathbf{P}(\boldsymbol{x} = y) - e^{-\epsilon}\mathbf{P}(\boldsymbol{x} = y - 1).$$

As per the proof, in the optimal distribution,

$$\alpha_y = \begin{cases} \delta & y = 0 \text{ and } Y > 0 \\ \delta\left(1 - e^{-2\epsilon}\right)e^{\epsilon y} & 0 < y < Y \\ T & y = Y \\ 0 & \text{otherwise} \end{cases},$$

where T is such that the total of all α_y is $1 - e^{-\epsilon}$.

The fact that the α_y must sum to $1 - e^{-\epsilon}$ can be proved by considering that each α_y contributes exactly $\frac{\alpha_y}{1-e^{-\epsilon}}$ probability to the distribution.

In order to prove conditions 5 and 6, we need to show

$$0 \leq T < \delta\left(1 - e^{-2\epsilon}\right)e^{\epsilon Y}.$$

In other words, α_Y should be the first α value that is lower than the geometric sequence of values given for the case $0 < y < Y$. Let us therefore compute the Y value for which this would be the case, in order to demonstrate that this is the value computed in Algorithm 6.

We can find the correct Y by noting that if $Y > 0$ then Y is the minimal value for which

$$\sum_{y=1}^{Y} \delta\left(1 - e^{-2\epsilon}\right)e^{\epsilon y} > (1 - e^{-\epsilon}) - \alpha_0.$$

Substituting in the sum for the geometric formula, we reach the Y value described in Line 3 of Algorithm 6.