SNS コミュニケーション分析手法を用いた 金融犯罪情報の早期検出に関する研究

大原　望乃† 　趙　智賢†† 　長田　繁幸†† 　中川　直樹†† 　小口　正人†

† お茶の水女子大学　〒 112–8610 東京都文京区大塚 2 丁目 1 番 1 号
†† 株式会社 日本総合研究所　〒 141–0022 東京都品川区東五反田 2 丁目 18 番 1 号
E-mail: †{g1920513,oguchi}@is.ocha.ac.jp, ††{zhao.zhixian,osada.shigeyuki,nakagawa.naoki}@jri.co.jp

あらまし　近年，フィッシング攻撃によるクレジットカード情報の窃取及び不正利用の被害は拡大しており，窃取されたカード情報の売買の場として SNS が多く利用されている．特にあるメッセージングアプリではカード情報売買を含む犯罪に悪用される事例が多数見られているが，先行研究により犯罪コミュニティの投稿に対してモニタリングを行うことで犯罪抑止効果があることが明らかになった．しかし，現行のシステムには監視者の個人情報が漏洩する可能性が存在する．本研究では，先行研究で使用されたモニタリングツールの実用化に即し，監視者の個人情報を秘匿する方法とそれを保証するための手段として，API による設定方法とプライバシサンドボックスを用いた監視者のプライバシの保護を提案する．実験から，手法の有効性を示し，さらに対象のアプリの脆弱性を発見した．

キーワード　セキュリティ，プライバシ，金融犯罪，ソーシャルメディア

1 はじめに

近年，多くの社会経済活動のデジタル化が進むに伴い，インターネットは我々の生活には欠かせないものとなった．そのような中，インターネットの普及と同時に犯罪もその場を変えサイバー空間上で行われることが増えてきている．

一般社団法人日本クレジット協会 [1] によると，表 1 のようにクレジットカード（以下，単にカードと呼ぶ．）の不正利用被害は年々増大し，2021 年通年の不正利用被害額は 330.1 億円にも上った．カードの不正利用被害は偽造カード被害，番号盗用被害，その他の不正利用被害と種類が分けられているが，そのうち番号盗用被害額は 311.7 億円と全体の約 9 割を占めており，被害額に占める構成比は増大傾向にある．

番号盗用の手口としてはフィッシング攻撃 [2] が挙げられる．フィッシング攻撃は主に電子メールの送信などによって偽装されたサイトに誘導しカード情報を窃取する，サイバー攻撃の 1 種である．昨今の不正利用被害の発生状況を鑑み，カード情報に対するフィッシング攻撃については効果的な対策の検討がなされてきた．

その 1 つとして，趙ら [3] の研究では，番号盗用の全体の流れのモデル化を行っている．これによると番号盗用による不正利用は単独で行われるものではなく，いくつかの過程に分類できる．さらにそれぞれの行為を別の人物あるいはグループが行っており，複数の人間による犯罪エコシステムが形成されている．

このシステムの中で，窃取されたカード情報が SNS 上で売買されているということが明らかになった．先の研究ではその特徴に注目した番号盗用の抑止方法の提案と，実際に実行した際に効果が見られたことが述べられている．しかし，この提案手法にはセキュリティ面での問題が確認された．本研究ではこの問題点に対する提案と実装を目的とする．

表 1　クレジットカード不正利用被害の発生状況

（単位：億円，%）

期間	クレジットカード不正利用被害額	クレジットカード不正利用被害額の内訳					
		偽造カード被害額		番号盗用被害額		その他不正利用被害額	
		被害額	構成比	被害額	構成比	被害額	構成比
2014年	114.5	19.5	17.0%	67.3	58.8%	27.7	24.2%
2015年	120.9	23.1	19.1%	72.2	59.7%	25.6	21.2%
2016年	142.0	30.6	21.6%	88.9	62.6%	22.5	15.8%
2017年	236.4	31.7	13.4%	176.7	74.8%	28.0	11.8%
2018年	235.4	16.0	6.8%	187.6	79.7%	31.8	13.5%
2019年	274.1	17.8	6.5%	222.9	81.3%	33.4	12.2%
2020年	253.0	8.0	3.2%	223.6	88.4%	21.4	8.5%
2021年	330.1	1.5	0.5%	311.7	94.4%	16.9	5.1%

本稿の構成は以下の通りである．第 2 章では先行研究と観察対象の SNS についての概要を紹介する．第 3 章で先行研究の手法の問題点を述べ，第 4 章でそれに対してプライバシサンドボックスの紹介と実現方法を提案する．第 5 章では提案手法の実装と実験の設計について述べる．第 6 章で実験結果とそれに対する考察を行い，第 7 章でまとめる．

2 先行研究

2.1 カード情報盗用手口のモデル化

趙らは，カード情報の窃取と盗用の全体の流れについて図 2 のようにモデル化を行った．（ここで，カード情報とはカードの会員番号や有効期限などをまとめて示すものとする．）

（1） 最初に，窃取の事前準備として誘導先のフィッシングサイトの作成や，誘導するための大量のメール送信，送信先となる電話番号やメールアドレスの販売がある．これらにより窃

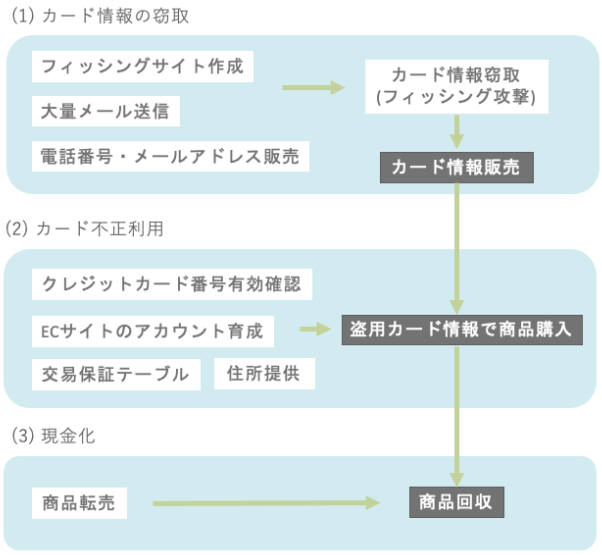


図 2　カード情報盗用の一連の流れ

取されたカード情報は，主に SNS を通じて販売される．

（2） 次に，盗用の手口として EC サイトにて換金性の高い品物の購入が行われる．このとき EC サイト側でもこのような行為を防止する対策がとられているため，アカウントの育成と販売も行われている．さらに専用の Web サイトで販売されているカード情報が有効かどうかの確認や，リスク低減のため第三者が売買を仲介する取引保証，購入した品物の受け取りに利用される住所の提供などの行為が存在する．

（3） 最後に，購入した品物の現金化が行われる．これは提供された住所に届いた品物の回収と，回収した品物の転売という行為に分業されている．

このように，番号盗用によるカードの不正利用には複数の人間の取引を行い協力することによって成り立つ犯罪エコシステムが存在している．しかし，この取引は違法性の高さから法的な効力を持つことはなく，彼らの間にある弱い信頼関係のみで成り立っている．

2.2　Telegram

先行研究では，カード情報が売買される SNS の観察対象に，Telegram（テレグラム）[4] を選択している．Telegram はテキスト，写真，ビデオなど様々なタイプのメッセージの送信や，音声電話，ビデオ電話などを行うことができるメッセンジャーアプリである．月間アクティブユーザ数は 7 億人を超え，世界で最もダウンロードされたアプリ 10 個のうちの 1 つに該当するなど世界規模で利用されている SNS といえる [5]．

Telegram は，その高いセキュリティ機能をセールスポイントの 1 つとしているが，一方でその匿名性の高さから犯罪に利用されやすいという一面を持っている．Telegram にはグループ機能 [6] という複数人でのチャット機能が備わっており，カード情報の売買がこのグループ機能を利用して行われている事例が観察されている．

2.3　モニタリングによる抑止とその効果

盗用被害の抑止方法として，前述した弱い信頼関係をさらに弱化させる方法が提案されている．SNS 上でのカード情報の売買の引き合いは互いの身元を隠した状態で行われるため，売買成立に必要な信頼関係を結びにくい．そこで売り手は信頼度を上げるためにカード情報の一部（例えばカード番号）をサンプルとして投稿する傾向にある．このときこの情報をクレジットカード会社が入手できれば，この時点でカードの利用停止などの対処を行い盗用被害を食い止めることが可能である．

さらに，提供された番号が利用不可能だったことで買い手から売り手に対する信用度は低下し，新規取引の機会の妨害にも繋がる．このようにして弱化した信頼関係は犯罪エコシステムの弱化を招き，その結果，盗用被害の抑止に繋がるといえる．

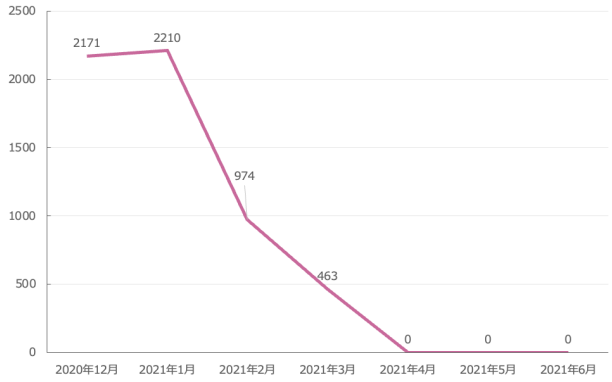


図 3　投稿されたカード番号の数

この研究では上記の抑止方法の有効性を確認するために，Telegram でカード情報売買に関する投稿を行っているグループを定期的にモニタリングして，カード情報が検知された場合はカード会社に通報するという実験を実施した．実際に実験対象としたグループの 1 つについて，投稿されたカード番号の推移の様子が図 3 である．このグループは元々1ヶ月に 2171 件のカード番号の投稿があったが，6ヶ月にわたって対策を行ったことでカード番号の投稿は 0 になった．さらに実験対象とした他の 2 つのグループについても，対策を開始してから数ヶ月でカード番号の投稿が 0 になっており，このモニタリングには一定の効果があることを示唆している．

3　問 題 提 起

3.1　先行研究のモニタリングツールの仕様

先行研究では図 4 のような設計のモニタリングツールが用いられている．これは大きく分けて 3 つのステップの手順からなる．

ステップ 1 では，まず，カード情報売買のために作成されたグループを検出するために，検索用のキーワードリストを作成する．Telegram では検索可能なパラメータはユーザ名かグループ名のみとなっており，チャットのテキスト内容などに関しては検索できない．そのため，ここではグループ名として使われ

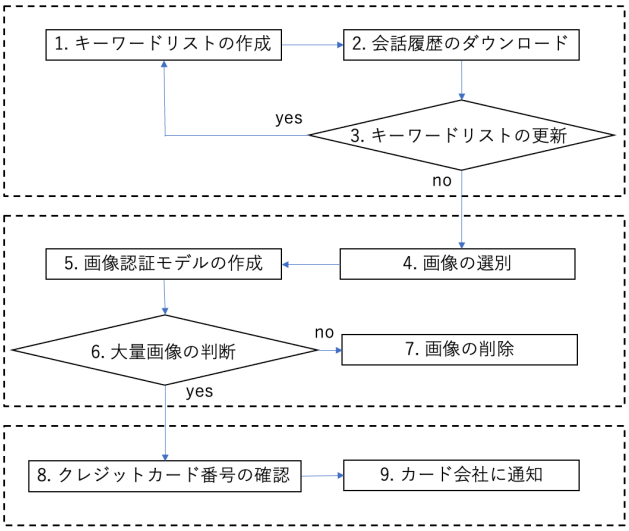


図 4　モニタリングツールの設計

ることが多い単語を収集する．次に，作成したキーワードリストを利用して Telethon[7] という Telegram が提供する API でグループの検索を行う．そして，検出したグループの会話履歴をダウンロードしテキスト情報と画像ファイルを入手する．ここで得られたテキスト情報を用いて，さらに番号盗用に関係する単語を特定し，キーワードリストを更新する．

ステップ 2 では，ステップ 1 で収集した画像ファイルについてカード情報が含まれるか否かを選別する．カード情報が含まれる画像には同じ桁数の数字が繰り返し現れるなどの特徴があるので，これを機械学習によって判別するようなモデルを作成する．さらに，これを用いて収集した画像ファイルすべてを選別する．このとき，カード情報が含まれないと判断された画像は破棄し，他方，含むと判断された画像ファイルは次のステップで処理する．

ステップ 3 では，得られた画像ファイルを目視確認して関係するカード会社へ連絡する．このとき，売り手が警戒しカード番号の一部が隠されている場合があるので，目視でカード番号の確認を行ったのちカード会社への連絡作業に移る．

3.2　セキュリティ上の問題点

前述したツールによるモニタリングは，前節の通り一定の成果を収めた．しかし，実験で監視に使用していた Telegram アカウントについては，登録していた電話番号の漏洩が起きてしまった．図 5 は漏洩した電話番号に対し DOS 攻撃 [8] を受けている様子である．

Telegram において電話番号はアカウントを作成するときに唯一入力が必要とされる要素であり，Telegram アカウントで設定必須な項目の中で最上位にあたる個人情報である．電話番号の公開範囲は 3 段階，すなわち，全員に公開する Everybody，連絡先に登録しているユーザのみに公開する My Contacts，非公開にする Nobody のいずれかを選択する．監視用アカウントでは公開範囲を Nobody に設定していたが，漏洩が発生した．モニタリングを安全に継続するために，監視者の電話番号を含

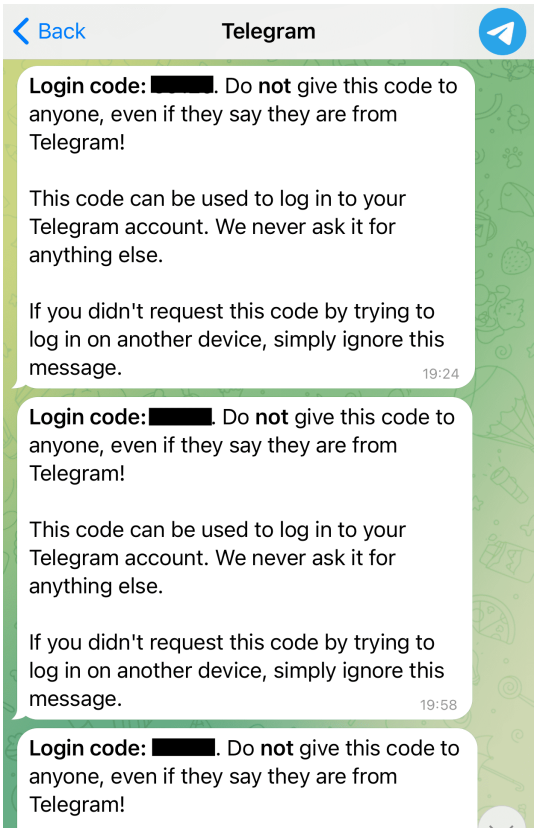


図 5　監視用アカウントが DOS 攻撃を受けている様子

む個人情報を秘匿できるような方法と，それを保証するための手段が必要である．

4　提　　案

前節で論じた 2 つの課題に対して，本研究では API によるユーザデータのパラメータの強制設定を行うという手法と，プライバシサンドボックスを作成してユーザのプライバシを観察するという手法を提案する．

4.1　API による強制設定

Telegram は API が豊富に存在するアプリケーションである．今までのモニタリングでは公開範囲を Nobody に設定していたにも関わらず，意図せず電話番号が漏洩していた．この際，公開範囲の設定はアプリ内の設定画面から行っていた．そこで，個人情報の秘匿化という課題については，ユーザ設定を強制変更するような API による解決を提案する．

具体的には，漏洩が問題となった電話番号の公開範囲の設定はユーザデータ内の InputPrivacyKeyPhoneNumber[9] というパラメータに該当する．このパラメータは API の 1 つ，account.SetPrivacyRequest[10] によりユーザが指定したものに変更することが可能である．すなわちこれを利用することで電話番号の公開範囲をアプリ内から変更することなく強制変更することが可能である．これは後述する秘匿の保証手段と合わせて個人情報の秘匿化に繋がると考える．

4.2 プライバシサンドボックス

秘匿化の保証が必要であるという課題に関しては，プライバシサンドボックスを用いたユーザのプライバシをモニタする手法を提案する．

Luo ら [11] によると，ソーシャルネットワークには個人情報の漏洩に関する様々な脅威が存在する．主に，以下のようなケースがある．

- 脈絡のない情報開示: コミュニティに信頼があると仮定して提供した情報が間違った設定やコードの誤動作，ユーザの誤解によって外部から簡単にアクセスできてしまうケース
- ネットワーク内の情報集約: メッセージの投稿ごとに少しずつ提供していた個人情報が全メッセージを集約することで大きなプライバシの損害になったケース
- ネットワーク越しの情報集約: リンクしている複数のソーシャルネットワークから情報を集約し意図しないプライバシの損害を生むケース

これらはいずれも自らが提供した個人情報について漏洩が起こっているケースである．

今回の個人情報の漏洩に関しては，漏洩の経路について現在も特定されていない．しかし，登録していた電話番号は実験用に取得したものであり，Telegram の登録以外には使用していないことから，漏洩の経路は Telegram を介したものに絞られる．さらにこの実験用アカウントはメッセージの投稿など自発的に情報を提供する行為を行っていなかったため，今回の漏洩は 1 つ目の脈絡のない情報開示にあたると推測される．

漏洩に関する脅威に対し，Luo らは個人情報とその開示のモデル化と，プライバシサンドボックスを利用したプライバシモニタの提案を行った．このモデルは以下の 3 つの定義を行っている．

定義 1. ネットワークの情報にどの程度アクセスできるかを示す指標として “openness level”（以下，OL) を用いる．例えば “OL = public” はすべての人がアクセス可能を意味する．同じ OL を持つネットワークは “access-equivalent グループ”（以下 AE グループ) に属する．

定義 2. 観察者は正直だが好奇心が強いモデル（Honest but curious observer）とする．すなわち，対象となるユーザについてできるだけ多くの情報を得ようとするが，ハッキングなど違法な行為は行わない．完全に合法（正直) だが非常に積極的に情報を求めている（好奇心が強い) モデルである．

定義 3. ソーシャルネットワーク N に提供されたユーザの個人情報データを $D(N)$ としたとき，観察者が得た個人情報データは $D'(N)$ である．このとき $D'(N) = D(N)$ であれば，観察者はユーザが開示しようとした情報を正確に見ており，ユーザのプライバシは保護される．しかし，$D'(N) - D(N) \neq \phi$ のとき，望ましくない個人情報の開示があるとみなす．

続いて図 6 に示したプライバシモニタのシステムの手順を簡略化して述べたものが以下である．

（1） 情報抽出: 観察者が N からユーザデータを取得する．

（2） 情報集約: 取得したすべてのデータから個人情報を集約し，$D'(N)$ を求める．

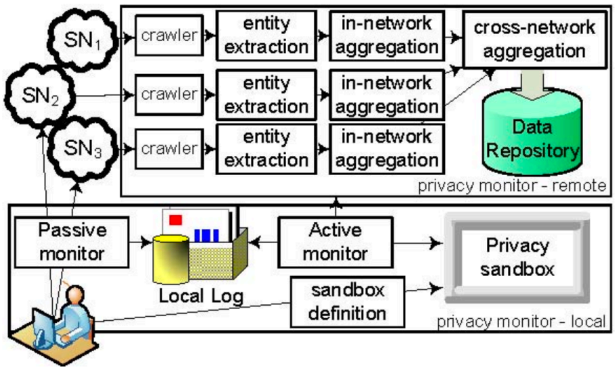


図 6 プライバシモニタのシステム構造 （出典: On Protecting Private Information in Social Networks: A Proposal）

（3） プライバシサンドボックス定義: ユーザは異なる AE グループに対して望ましい $D(N)$ を定義する．

（4） アクティブモニタ: $D(N)$ と $D'(N)$ の違いを発見した場合，ユーザに対して警告を出す．ユーザは，プライバシサンドボックスの設定を変更するか，個人情報の一部を失効させることで問題に対処することができる．

この提案法ではプライバシサンドボックスとはユーザのプライバシプランを管理するものと定義されている．すなわち，ユーザが期待している $D(N)$ と実際の $D'(N)$ のギャップをローカルで管理し，それをユーザが認識することで情報漏洩の脅威に対処するという仕組みを指している．

4.3 実 現 方 法

4.1. 節を踏まえた上で Luo らの提案法を我々の研究に拡張すると，以下の手順で秘匿化と保証を実現する．

準備: 観察者，対象ユーザとして 2 つの Telegram アカウントを用意する．

（0） 対象ユーザは API を利用して個人情報の公開範囲を設定する．

（1） 情報抽出: 観察者が API を利用して対象ユーザのユーザデータを取得する．

（2） 情報集約: 取得されたデータについて個人情報を集約し，$D'(N)$ を求める．

（3） プライバシサンドボックス定義: 対象ユーザは観察者（すなわち，実際の場合は犯罪コミュニティ）に対して望ましい $D(N)$ を定義する．その一部には登録済の電話番号の公開範囲が Nobody であるという条件を含む．

（4） アクティブモニタ: $D(N)$ と $D'(N)$ の違いを発見した場合，対象ユーザに対して警告を出す．対象ユーザは，Telegram 上の設定を変更し問題に対処することができる．

5 実装と実験設計

5.1 実　　装

また $D'(N)$ は，観察者が API の users.GetFullUserRequest[12] で得られるユーザデータすべてに設定した．さらに，このとき

得たユーザデータはローカルで随時保存しておき，最新のものを $D(N)$ として採用する．すなわち，アクティブモニタによる $D(N)$ と $D'(N)$ の比較は users.GetFullUserRequest クエリの最新結果と 1 つ前の結果を比べていることになり，アクティブモニタはユーザデータに起きた何らかの変化をすべて検知し通知するという仕様となる．

$D(N)$ と $D'(N)$ の比較については，取得したユーザデータをローカルのテキストファイルに保存し，ファイル同士を比較するプログラムを実装した．比較の結果はログファイルに記録され，特に違いがなれけば “No change.”，違いがあれば異なる箇所のみを抽出して出力する．

実装には Telethon を用いたほか，手順の繰り返し実行には cron を利用した．

5.2 実験設計

まず，新たに 3 つの Telegram アカウント A，B，C を用意した．アカウント A，B，C はそれぞれ異なる場所とデバイスで動作するものとし，それぞれ表 7 で示すような役割が与えられている．

表 7　モニタリング自動化のテスト環境

アカウント	実験におく役割
A	グループ A’ で定期的に発言
B	グループ A’ の会話履歴をダウンロード グループ検索を定期的に行う
C	アカウント B のユーザ情報を監視 グループ A’ の情報を監視

アカウント A は仮想の犯罪コミュニティの役割を担う．A は，はじめにグループ A’ を作成したのち，グループ A’ で定期的にカード情報の販売宣伝に見立てたメッセージと画像ファイルを投稿する．この動作は毎時 0，10，20，30，40，50 分（10 分ごと）に自動的に行われる．

アカウント B は犯罪コミュニティの監視者の役割を担う．B は，はじめに API により公開範囲を強制設定したのち，グループ A’ の会話履歴のダウンロードとキーワードからのグループの検索を定期的に行う．会話履歴のダウンロードは毎時 0，5，10，15，20，25，30，35，40，45，50，55 分（5 分ごと）に自動的に行われ，グループ検索は毎時 0，10，20，30，40，50 分（10 分ごと）に自動的に行われる．

アカウント C は監視者のプライバシの観察者の役割を担う．C は，アカウント B のユーザ情報と，グループ A’ の情報の監視を定期的に行う．ユーザ情報の監視は毎時 0，20，40 分（20 分ごと）に自動的に行われ，グループ情報の監視は毎時 10，30，50 分（20 分ごと）に自動的に行われる．

またこの実験では，平常状態と個人情報が漏洩している状態を両方検知できるかテストする目的で，定期的に監視者の電話番号の公開範囲設定を Everybody に変更する動作を行った．そしてその前後のアクティブモニタの動きを観察した．

6 実験結果

6.1 実験結果

実験期間中は 20 分ごとに正常に比較プログラムが動作し，都度 20 分前からの変化状況がログに記録された．ログの見方は，データに変更があった場合，変更箇所について冒頭にマイナス記号がある行が以前のデータ，プラス記号がある行が最新のデータを示している．

1. 個人情報の漏洩が発生していない状態

アカウント B のユーザ情報について観察を記録したログを図 8，グループ A’ の情報について観察を記録したログを図 9 に示す．このときアカウント B については会話履歴のダウンロードとキーワード検索のみを行っている状態であり，ユーザ情報の変更に繋がる行為は行っていない．したがってログにも変化なしを示す “No change.”が繰り返し記録されている．一方グループ A’ については定期的にメッセージと画像ファイルを投稿している状態である．ログには “pts”というパラメータの数字が 4 ずつ増えていく様子が記録されており，このパラメータがグループの投稿数に該当すると推測される．

```
2022-12-29 09:20:02.119570
No change.
2022-12-29 09:40:01.570037
No change.
2022-12-29 10:00:01.838666
No change.
```

図 8　アカウント B のユーザ情報のログ（漏洩なし）

```
2022-12-29 09:10:01.534767
-     pts=5855,

+     pts=5859,

2022-12-29 09:30:02.264178
-     pts=5859,

+     pts=5863,

2022-12-29 09:50:01.669256
-     pts=5863,

+     pts=5867,
```

図 9　グループ A ’ の情報のログ（漏洩なし）

2. 個人情報の漏洩が発生した状態

監視者の電話番号の公開範囲設定が Everybody に変更された状態である．アカウント B のユーザ情報について観察を記録したログを図 10 に示す．グループ A’ については，図 9 と同じ結果となったため省略する．公開範囲設定の変更は 10 時に行われた．このときログには少しのラグを経て “add_contact”のパラメータが False から True に，“phone”のパラメータが None

図 10　アカウント B のユーザ情報のログ（漏洩あり）

から登録していた電話番号に変化したことが記録された．公開範囲設定により保護されていた電話番号が設定の変更により漏洩したことがわかる．さらに電話番号が公開されたことで連絡先への追加が自動的に許可されたことも読み取れる．

3. 2. を経て個人情報の漏洩が発生していない状態

監視者の電話番号の公開範囲設定が Nobody に変更された状態である．アカウント B のユーザ情報について観察を記録したログを図 11 に示す．グループ A' については 2. と同じく図 9 と同じ結果となったため省略する．公開範囲設定の変更は 11 時に行われた．このときログには 2. とは逆に “add_contact” のパラメータが True から False に，“phone” のパラメータが登録していた電話番号から None に変化したことが記録された．つまり公開範囲設定の変更により，漏洩していた情報が保護されたのが確認できる．これは提案手法によって個人情報の秘匿化が成功していると言える．

図 11　アカウント B のユーザ情報のログ（再び漏洩なし）

以上の結果より，実装したシステムは個人情報の漏洩状態について正確に示した．このシステムはプライバシサンドボックスとして機能しており，期待した役割を十分に担っている．

6.2　予備実験での結果

上記の実験ではユーザデータの観察は 20 分ごとに行われたが，実験環境を設定するにあたり観察の間隔を 1 分に縮めた状態で予備実験を行ったところ，図 12，図 13 のような記録が得られた．

このとき公開範囲設定の Everybody から Nobody への変更

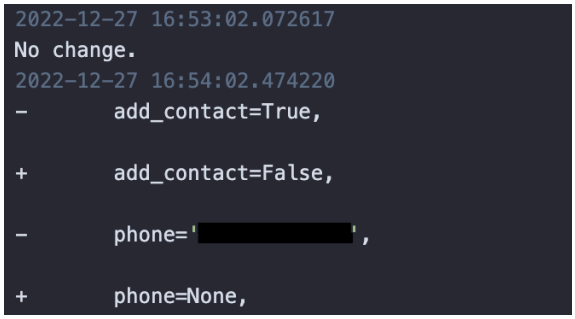


図 12　予備実験にて公開範囲を Nobody に変更した時刻のログ

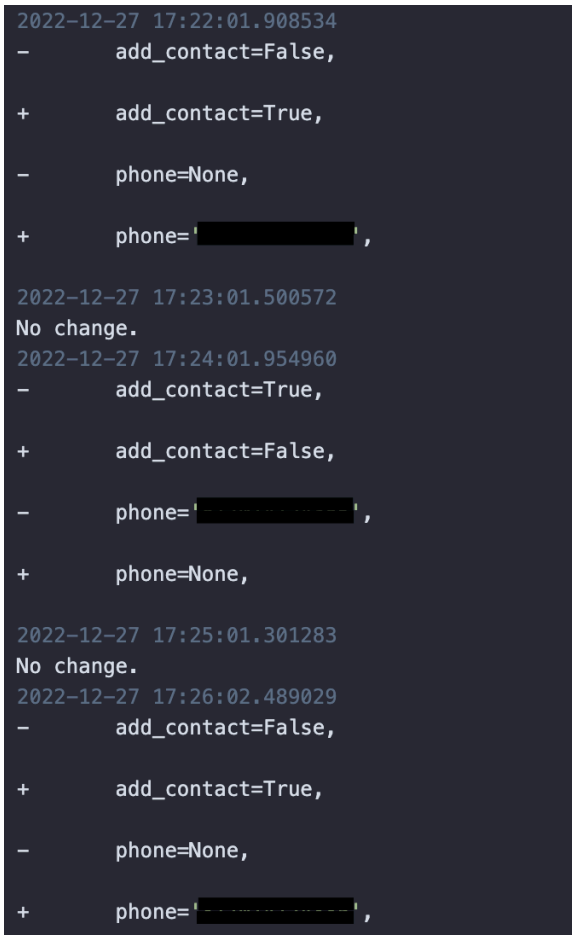


図 13　図 12 の後のログ

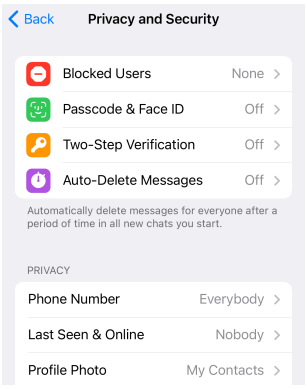
は 16 時 53 分に行われた．図 12 の記録にもその時刻の直後に各パラメータが変化した様子が残っている．しかし記録を辿っていくと，図 13 のように 17 時 22 分頃から電話番号が公開と非公開を繰り返していることが明らかになった．この現象は最終的に 17 時 30 分に非公開になったのを最後に止まった．

これを受け，何故このような現象が起きるのか発生条件を探ったところ，以下の条件と手順で変更したはずの設定が反映されない現象が起きることを特定した．

条件: 観察対象のアカウントが Telegram の携帯アプリと PC アプリ両方から同時にログインされていること．

手順:

1. Telegram の携帯アプリと PC アプリ両方でセキュリティ設定画面（図 14 または図 15）を開く．



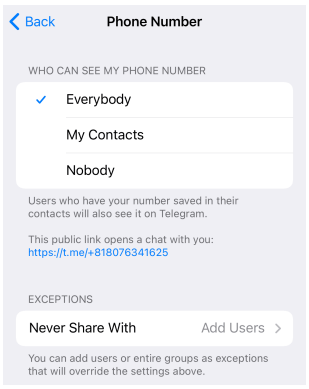


図 14　図 15 Telegram のセキュリティ設定画面

2. 片方のアプリで公開範囲を Everybody から Nobody に変更する．もう一方のアプリは画面を動かさず設定画面を維持する．
3. 変更を行なったアプリをセキュリティ設定画面以外の画面に切り替える，またはアプリを閉じる．
4. 変更を行なったアプリで再びセキュリティ設定画面を開く．

すると，Nobody に設定したはずの公開範囲が Everybody に戻っているのが確認できる．

これは，複数の端末の設定ファイルがサーバ側で同時に同期される場合，端末ファイルの中身が違ったときのエラー処理がうまくできていないということが考えられる．予備実験での現象も，API で設定を変更している間にアプリでセキュリティ設定画面を開いた状態になっていて，同期状態が不安定だったことにより引き起こされた可能性は高い．

この予備実験の結果は，注意しないと意図せぬ情報漏洩が起きる可能性があることを示唆している．プライバシサンドボックスがあればプライバシが漏れているかどうか把握できるため，情報漏洩の可能性を抑える効果があると改めて主張する．

7　ま　と　め

先行研究で発生したモニタリング時に監視者の個人情報の漏洩の可能性がある問題について，監視者の個人情報の秘匿を目的として，API による強制的な設定方法を，さらにその保証として，プライバシサンドボックスを用いた監視者のプライバシのモニタを提案した．実験では 2 つの手法を組み合わせ，個人情報の漏洩を防ぎながらモニタリングを行うシステムを構築し，平常状態と漏洩が発生している状態の比較を行った．その結果，構築したシステムが漏洩を検知できていることを確認した．提案手法は期待した役割を担っていると考えられる．

さらに，予備実験の結果より，Telegram でユーザ設定をするとき特定の操作を行うことで設定が無効化されることを発見した．この結果は意図しない情報漏洩が起きる可能性を示しており，プライバシサンドボックスの必要性を示している．

文　　献

[1] 一般社団法人日本クレジット協会, クレジットカード不正利用被害額の発生状況, https://www.j-credit.or.jp/information/statistics/download/toukei_03_g.pdf （2023 年 1 月参照）
[2] 総務省, 国民のための情報セキュリティサイト フィッシング詐欺に注意, https://www.soumu.go.jp/main_sosiki/cybersecurity/kokumin/enduser/enduser_security01_05.html
[3] 趙 智賢, 長田 繁幸, SNS を経由するクレジットカード不正利用のモデル化と抑止方法の検討, 研究報告セキュリティ心理学とトラスト（SPT）, 2022-SPT-48, No.25, pp.1-7, 2022.
[4] Telegram, https://telegram.org/
[5] Telegram FAQ What is Telegram?, https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here
[6] Telegram FAQ Groups and Channels, https://telegram.org/faq#groups-and-channels
[7] Telethon, https://tl.telethon.dev/
[8] IPA, ネットワークセキュリティ関連用語集 “DoS attack”, https://www.ipa.go.jp/security/ciadr/crword.html#D
[9] InputPrivacyKeyPhoneNumber, https://tl.telethon.dev/constructors/input_privacy_key_phone_number.html
[10] SetPrivacyRequest, https://tl.telethon.dev/methods/account/set_privacy.html
[11] Bo Luo, Dongwon Lee, On Protecting Private Information in Social Networks: A Proposal, 2009 IEEE 25th International Conference on Data Engineering, pp.1603-1606, 2009
[12] GetFullUserRequest, https://tl.telethon.dev/methods/users/get_full_user.html