Nozioni per progetto DPP

martedì 19 dicembre 2023 14:13

G → rich interation graph

V → set of entities
per esempio, in un OSN (social network) V può rappresentare l'insieme degli utenti.

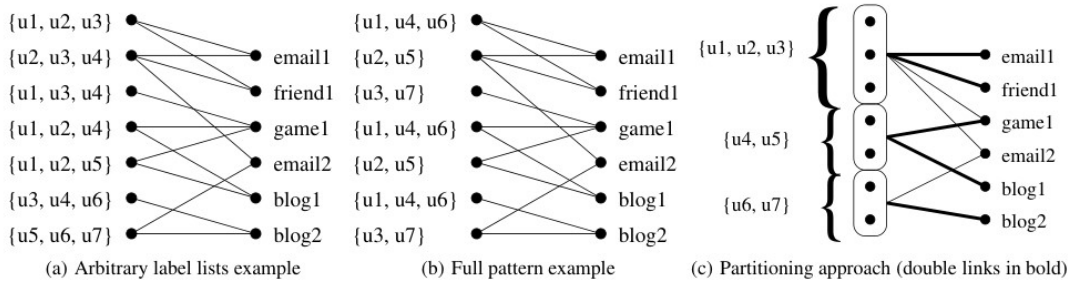
I → sottoinsieme di V che contiene nodi che corrispondono a interazioni tra nodi di V
Interazioni → invio di una mail, invio di un messaggio, un game tra giocatori, pubblicazione di un blog ecc.
Per rappresentare le interazioni usiamo un ipergrafo (un arco può essere fra più di due vertici)

E → insieme degli archi tra nodo di V e nodo di I

Sia G il nostro grafo, il nostro obiettivo è creare una versione anonimizzata G'

x(v) → mappatura del nodo v che nasconde i dati nel grafo anonimizzato G'

Innanzitutto lascio la foto dei tipi di approccio che vedremo:



(a) Arbitrary label lists example (b) Full pattern example (c) Partitioning approach (double links in bold)

Figure 2: Examples of different anonymization approaches

anonimizzazione tramite label lists: l'ID di ogni nodo viene sostituito da una lista di possibili ID (o etichette). Così riveliamo delle informazioni parziali circa il mapping dai nodi al loro vero identificatore e i link tra di essi.

L(v) -> funzione da V a P(x), cioè creazione dell'elenco di possibili etichette di V.
x(v) appartiene a L(v) (ricordiamo che x(v) è il vero identificatore di V).

Siccome la semplice scelta di liste non strutturate è vulnerabile possiamo suddividere i nodi in gruppi più piccoli o usare altre tecniche.

Arbitrary list approach --> lista non strutturata, creata arbitrariamente, quindi debole (metodo più stupido in poche parole), poiché possono capitare situazioni in cui capiamo le interazioni di una buona parte di utenti andando "ad esclusione" con le informazioni a nostra disposizione.

Uniform list approach --> approccio che genera un insieme di liste più strutturate che evitano attacchi statici mantenendo la privacy anche quando l'hacker ha conoscenza di base sul grafo (in poche parole guardando lo schema così com'è non può trovare informazioni utii al

ricreare lo schema orignale).
Per garantire la privacy ci deve essere abbastanza diversità tra le interazioni dei nodi della stessa classe.

Quindi si parla di "**proprietà di sicurezza delle classi**" se ogni nodo v, dopo la separazione in classi, partecipa ad interazioni di ogni tipo con al massimo un nodo per ogni classe.

Tipo di approccio **greedy** --> Prendiamo il nodo v e lo mettiamo nella prima classe che ha meno di m membri, guardando che non violi la condizione di sicurezza (quindi garantendo che ogni nodo che partecipa ad un'interazione con v non partecipi ad una interazione con un nodo già presente nella classe).
Questa operazione si chiama SAFETYCONDITION e può essere implementata mantenendo per ogni classe una lista di nodi che hanno già un'interazione con qualsiasi altro membro della classe.
L'inserimento è sicuro se per ogni w (= nodo che ha interazione con v) si che che v e w non sono presenti in questa lista.

Un'operazione da fare è ordinare i dati in base ad una caratteristica utile per il nostro carico di lavoro e mettere insieme nodi che hanno valori simili (per esempio tutti abitanti del giappone), in modo che a seguito di query ci sia ancora più possibilità di anonimizzazione, siccome tra un utente e l'altro è meno probabile trovare differenze che aiutino l'hacker a trovare l'identità dell'utente.

Perciò si mettono gli attributi in ordine di importanza per il carico di lavoro e si ordina in base a tutti questi valori. Poi attuiamo l'inserimento greedy sul dataset ordinato.
Così si ha molta più possibilità di mettere i nodi simili nella stessa classe o in classi vicine (sempre che non intacchino la sicurezza).

Ora, dopo aver diviso i dati in classi di dimensione m, generiamo label lists da attribuire ad ogni nodo per ogni classe.
Usiamo un metodo simmetrico per generare ste liste, tramite un parametro k ed un pattern p.

→ Data la classe C con m entità, formiamo una lista di m liste basata su un pattern intero p = {p0, p1, … , pk-1}, che è sottoinsieme di {0, 1, … , m-1} ma con dimensione k.
Iterando per i che va da 0 ad m creiamo le varie liste di labels per ogni entità u0, u1, … , um-1:

$$\text{list}(p,i) = \{u_{i+p_0 \mod m}, u_{i+p_1 \mod m}, \ldots u_{i+p_{k-1} \mod m}\}.$$

Questa definizione la si usa perché è simmetrica, cioè c'è un unico schema che si ripete ciclicamente per generare le varie liste e mantiene la proprietà di sicurezza

Esempio mezzo a caso di applicazione:

Uniform List Example. Given entities $u_0, u_1, u_2, u_3, u_4, u_5, u_6$ and the pattern $0, 1, 3$, we form label lists to assign to nodes as:

$\{u_0, u_1, u_3\}$ $\{u_1, u_2, u_4\}$ $\{u_2, u_3, u_5\}$ $\{u_3, u_4, u_6\}$
$\{u_4, u_5, u_0\}$ $\{u_5, u_6, u_1\}$ $\{u_6, u_0, u_2\}$ □

Un esempio di pattern usato è il **prefix pattern**, cioè quando p = {0, 1, 2, … , k - 1}. In quest caso si mantiene una linearità simmetrica che facilità l'analisi successiva.

Un altro esempio è il **full pattern**, cioè come prima ma con k = m. in questo modo ogni label

list nella classe è identica, e contiene tutti i nodi di quella classe.

La notazione *(k,m)-uniform list* si riferisce a liste generate su classi di dimensione almeno m ed un pattern di dimensione k.
La notazione *(k,k)-uniform list* invece indica la lista completa.
Si parla di prefix list per indicare il caso in cui creiamo liste con un prefix pattern in cui si ha k < m.

I due parametri k ed m devono sottostare ad un tradeoff tra utilità e privacy.

Per esempio una lista (1, 1) associa ogni nodo direttamente con l'entità corrispondente, consentendo la piena utilità ma per niente la privacy.
Invece una lista (|V|, |V|) associa ogni nodo con la lista di tutte le etichette possibili, e rappresenta quindi minima utilità ma massima privacy.

La scelta dei parametri k ed m quindi dipende dai dati e dal grado di privacy desiderato.

Quindi k corrisponde alla dimensione dei gruppi nell'anonimizzazione dei dati delle tabelle (vedi k-anonimità).
Il parametro m permette invece un range più ampio di possibili anonimizzazioni da poter fare.

Quindi per un fissato k, ogni m >= k può essere scelto. Una anonimizzazione di tipo (k,k) genera k! Possibili soluzioni, mentre (k,m) genera più possibili soluzioni, quindi un diverso tradeoff tra utilità e privacy.
Per esempio: (3,3) dà 6 possibilità, mentre (3,4) dà 9 possibilità.

Ora bisogna assegnare ad ogni nodo una lista di etichette, e bisogna stare attenti di dare ad ogni nodo una lista che contenga la sua etichetta originale.
Bisogna evitare gli schemi ovvi e prevedibili, perche se l'hacker capisce la mappatura può decodificare tutto.

Possiamo vedere il procedimento come un problema di corrispondenze di un grafo bipartito con m nodi su ogni lato: m entità ed m liste da assegnarvi.
Ogni arco deve collegare un nodo a una lista che contiene la sua vera etichetta.

L'obiettivo è creare l'insieme di m archi che non abbiano alcun vertice in comune tra loro.
Ci serve un metodo randomico/arbitrario, quindi partiamo da un nodo a caso, gli diamo una lista a caso (controllando che contenga il suo ID), eliminiamo gli archi che toccavano i due vertici appena collegati e passiamo ad un altro.

> TEOREMA 1. *Un aggressore che osserva i dati pubblicati utilizzando l'approccio della lista uniforme* (k, m) *e che non ha alcuna conoscenza dei dati originali può indovinare correttamente quali entità partecipano a un'interazione con una probabilità massima di* $1/k$.

Quindi il cattivone non può fare molta inferenza sul processo adottato. Inoltre vediamo che le label lists sono sicure anche contro certi tipi di conoscenze di base, cioè se ad esempio un cattivone sa identificare un po' di utenti:

TEOREMA 2. *Un aggressore che osservi i dati pubblicati utilizzando un elenco completo e sia in grado di utilizzare le conoscenze di base per trovare [redacted]*
L'identità di almeno r nodi può indovinare quali altre entità partecipano a un'interazione con una probabilità pari a $1/(k - r)$.

Da tenere presente solo che se un bro sa la vera identità di un utente ed i dati anonimizzati usando il (k,m)-prefix pattern, può vedere quali interazioni ha tale utente (per esempio quante mail ha mandato e quanti amici ha nella lista), ma non può ricavare informazioni precise su tali dati, cioè non può sapere a chi ha mandato tali mail e neanche chi sono gli amici nella sua lista.

Inoltre gli autori ci fanno capire che tali assunzioni sono praticamente impossibili nella realtà, poiché stiamo assumendo che il cattivone abbia a disposizione informazioni su gente della stessa classe (sarebbe una coincidenza della madonna!)

Degli standard che si scelgono per il valore di k sono 5 o 10, ma se si vuole molta sicurezza si può anche andare tra 20 e 50.

Siccome comq il metodo ha delle debolezze date da possibilità di conoscenze di base degli hackers oppure da strutture deboli (per esempio in casi in cui ogni utente, che so, ha un solo amico in ogni stato) e quindi vulnerabili a reverse decoding, si può pensare ad altri metodi rispetto alle liste di labels.

Dobbiamo cercare un metodo che impedisca attacchi basati sulle conoscenze di base, quindi aumentiamo la quantità di mascheramento dati, anche se ciò rimuove un po' di utilità.

L'approccio usato sarà un approccio di partizione, che suddivide quindi i nodi in classi.
Le informazioni sugli archi non saranno presenti ma sarà presente il numero di archi all'interno di ogni sottoinsieme.

Dato un grafo G ricco di interazioni, una partizione anonima di G è un insieme C di nodi che prende un sottoinsieme dei vertici di V. La partizione anonima è un grafo pesato bipartito G' su C ed I in modo tale che il peso dell'arco (C, i) è il numero di archi tra i nodi della classe C ed una interazione i.

Si ha la stessa condizione di sicurezza dell'altro approccio: anche se i link tra i nodi e le interazioni non sono rivelati dalla partizione, la condizione di sicurezza serve per impedire che il cattivone sfrutti la densità del grafo per capire quali nodi partecipano in una certa interazione con alta probabilità.

COROLLARY 1. *An attacker who observes data published using the m-partition approach and who has no background knowledge about the original data can correctly guess which entities participate in an interaction with probability at most* $1/m$.

THEOREM 3. *An attacker with background knowledge about interactions of an entity, modeled as knowing the true identity of some nodes, and the fact that these nodes participate in certain interactions, and data anonymized into an m-partition with the safety condition can correctly guess which entities participate in interactions about which nothing is known with probability at most* $1/m$.