Pedro Ivo Siqueira Nepomuceno

Preservando a privacidade individual na análise de dados de domínio público

São Paulo
Novembro de 2017

Pedro Ivo Siqueira Nepomuceno

Preservando a privacidade individual na análise de dados de domínio público

Trabalho de Conclusão de Curso para obtenção de nível superior do Bacharelado em Ciência da Computação no Instituto de Matemática e Estatística da Universidade de São Paulo (IME - USP)

Orientador: Prof. Dr.Vladmir Belitsky

São Paulo
Novembro de 2017

Agradecimentos

Aos professores que me ajudaram a chegar até aqui, à minha família por fazer com que fosse possível chegar e permanecer na USP, e aos meus amigos, por tornar essa jornada muito maior do que um simples diploma.

" I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference."
(Robert Frost, The Road not Taken,1916)

Resumo

Impulsionada por dados, a ciência pode auxiliar a extrair diversas informações acerca de uma população, podendo inclusive prever tendências e indicar o melhor caminho a se seguir. Seu avanço, entretanto, acaba esbarrando em um problema legal: a privacidade individual. Muitos dos dados que poderiam ser usados por cientistas (ou até corporações privadas) são protegidos por força de lei.

É neste contexto que surgiram os bancos de dados estatísticos: bancos que respondem somente a consultas que revelem dados agregados, sem distinção individual dos registros. A simples retirada dos dados de identificação individual, entretanto, não garante o anonimato. Utilizando técnicas de programação linear ou de força bruta, é possível identificar cada registro do banco por meio de sucessivas consultas de agregação. Para impedir essa identificação, os bancos passaram a responder as consultas aliadas a uma perturbação aleatória que visa impedir a aplicação de tais técnicas.

O presente projeto tem por objetivo o estudo sobre estes bancos, as técnicas que são utilizadas para garantir a privacidade, e principalmente, um limite inferior que precisam ter as perturbações aleatórias para não serem completamente inúteis. O trabalho apresenta justificativas de um limite inferior de $o(\sqrt{n})$ para as perturbações.

Além disso, inclui a criação de um simulador para verificar os tempos de execução das decriptações, a sua assertividade e mais dados que auxiliem na compreensão do leitor aos bancos estatísticos.

Palavras Chave: Banco de Dados; Privacidade;Perturbação;Segurança;Bancos Estatísticos

Abstract

Powered by data, science is capable of extarcting many information about a population. It is even possible to indicate the best path to follow when solving a problem. However, its advance is delayed by privacy issues. Many useful data is protected by privacy laws.
In this context, statistical databases arose. This are databases that only answer aggregate data queries (such as sum, average or standard deviation). It does not reveal individual rows data. Nevertheless, separating names and IDs from sensitive information does not guarantee anonymity. Using linear programming or brutal force techniques, it is possible to identify wach row of data through successive aggregation queries. In order to secure identification, databases started to only answer queries allied to random noise.
The purpose of this project is study this kind of database, the techniques to guarantee privacy and obtain lower bounds to the necessary noise to secure the database. This project justifies a lower bound of $o(\sqrt{n})$ noise.
Additionally, it includes the creation of a simulatorto verify decryption algorithms execution time, assertiveness and more data tha could help the reader to understand statlimite inferioristical databases.
Keywords: Database; Privacy;Noise;Security;Statistical Databases

Sumário

1 Introdução

O interesse pelo tema deste projeto surge de um assunto mais genérico que a privacidade de dados em si. O presente projeto teve seu início no estudo da geração de grafos a partir de dados sensíveis como prontuários médicos de pacientes do sistema público de saúde ou as relações desenvolvidas por indivíduos em uma rede social. A geração destes grafos possibilitaria que estudos sobre, por exemplo, os padrões de crescimento, de formação e esparsidade fossem esclarecidos. Por se tratar de dados sigilosos, estes estudos esbarram na problemática da manutenção da privacidade.

A simples remoção da identificação dos donos destes dados não garante a manutenção da privacidade. Estudos demonstraram[7] que esta técnica pode ser facilmente burlada por algoritmos simples, que podem re-identificar os nós da rede por meio dos outros atributos que não são ocultos. Seria equivalente, por exemplo, a mostrar os dados acadêmicos dos alunos de uma universidade escondendo os seus nomes. A simples informação da idade, do ano de ingresso e do curso de ingresso poderiam ser suficientes para identificação de boa parte (se não a totalidade) dos estudantes.

É neste contexto que *Dwork et al* desenvolveu, em 2012, um estudo sobre a geração de grafos que possuíssem as mesmas características estatísticas que a população estudada, respeitando contudo um nível de proteção chamado de privacidade diferencial.[4] Tais grafos são gerados utilizando um objeto chamado *Graphon*. Este objeto é obtido através da população de origem, de maneira que a retirada de um único nó não reflita na sua geração. Com ele, seria possível gerar outros grafos que seguem a mesma distribuição do original, sem permitir identificar indivíduos isolados.

O material de estudo sobre essa problemática é, entretanto, denso e cheio de conceitos escondidos profundamente nas suas referências. E para entender o tema maior, é necessário atentar-se primeiro aos detalhes das notações utilizadas e dos conceitos que sustentam suas teses. O presente projeto tem como objetivo apresentar ao leitor as ferramentas essenciais para o entendimento da privacidade de dados, de consultas estatísticas e da preservação desta privacidade após sucessivas consultas. Para isso, serão apresentados as ideias do artigo que iniciou toda a linha de pesquisa sobre privacidade diferencial[3] da maneira mais didática e clara para o leitor quanto seja possível.

O desenvolvimento deste trabalho conta também com um simulador, escrito em Python, para realização de testes do algoritmo exemplificado por *Nissim et al.*, além de alguns outros testes originais do presente texto.

2 Formulação do Problema

2.1 Motivação Inicial

O trabalho consiste na análise de privacidade de um *banco de dados estatístico*. Define-se como tal um banco que não permite todos os tipos de consultas. Permite somente consultas agregadas, que possam dar uma idéia da distribuição dos dados àquele que o está consultando [9]. Pode-se incluir como exemplo de consultas agregadas: a soma de componentes do banco, o máximo componente dentre aqueles selecionados, o mínimo, média, variância, etc.

Os referidos bancos possuem proteções que impedem que quem o esteja consumindo identifique facilmente os registros individuais. Pode-se imaginar, por exemplo, um banco de dados que armazene as transações internacionais de contribuintes brasileiros residentes fora do país. Para pesquisadores e analistas de mercado, obter informações de assiduidade, quantidade de dinheiro por remessa e as datas de maior movimentação pode representar um material valioso para o trabalho que desenvolvem. Não pode ser permitido, entretanto, que estes contribuintes sejam identificados. O sigilo sobre cada transação e sobre quem está transferindo e para quem deve ser mantido. Outro possível exemplo são dados de prontuários médicos. Para a ciência, dados que possam esclarecer a maneira como ocorre a transmissão de uma epidemia representa grande valor. É garantia legal dos pacientes, entretanto, ter a sua identidade preservada.

Diversos tipos de proteções podem ser aplicadas. O texto motivador original não seleciona nenhuma das técnicas especificamente, falando somente em limites inferiores e superiores para a proteção. Na verdade, a técnica escolhida em si é indiferente, já que matematicamente, funcionam da mesma forma: adicionando uma perturbação aos resultados da consulta, de maneira direta ou indireta.

2.2 Introdução a notação utilizada

O desenvolvimento deste texto se remeterá a alguns conceitos que serão vastamente utilizados até sua conclusão. Antentar-se à notação utilizada e às definições mais básicas é peça chave para interpretação clara.

Uma variável binária é tal que assume somente dois valores, comumente representados por True e False, Sucesso e Fracasso, ou mais difundido: 0 ou 1. Para os efeitos deste trabalho, um banco de dados (denotado por d) é um vetor de dimensão n, com cada uma de suas componentes sendo uma variável binária. Temos então:

$$d = (d_1, d_2, ..., d_n), d_i \in \{0, 1\} \tag{2.1}$$

um banco de dados. Ou ainda:

$$d = \{0, 1\}^n \tag{2.2}$$

Note que os conceitos de perturbação e auditoria de consultas apresentados anteriormente podem ser aplicados a estes bancos de dados, assim como bancos que não fossem binários.

Denotamos por $\mathbb{P}$ a medida uniforme de probabilidade no espaço $\{0, 1\}^n$, e por Q uma função aleatória em $\{0, 1\}^n$ cuja distribuição segue a medida $\mathbb{P}$. Assim, temos que:

$$\forall q \in \{0, 1\}^n, \mathbb{P}[Q = q] = \left(\frac{1}{2}\right)^n \tag{2.3}$$

De maneira mais simplificada, isto significa que **cada q tem a mesma probabilidade de ocorrer que qualquer outro q'**.

A variável aleatória Q será referida como uma *consulta aleatória* no banco. As consultas no banco são posições sobre as quais serão obtidas informações no banco de dados. A resposta exata a essa consulta é a soma dos valores dessas posições no banco de dados:

$$\sum_{i \in Q} d_i \tag{2.4}$$

a resposta exata de uma consulta Q.

Como já citado, a perturbação de consultas será utilizada pelo banco para atingir a privacidade no nosso simulador. Qualquer outro tipo de perturbação, entretanto, encaixa perfeitamente nas definições deste texto. Definamos então a variável aleatória R, e Pr a sua medida de probabilidade associada. Note que $\mathbb{P} \neq Pr$. Definimos ainda que R é limitada por uma função determinística $\mathcal{E}, \mathcal{E} : \mathbb{N} \to \mathbb{R}^+$, de maneira que:

$$|R| \leq \mathcal{E}(n) \tag{2.5}$$

Para cada consulta Q_i ao banco de dados é gerado um ruído R_i associado. Como última definição desta seção, definimos a *resposta com ruído aleatório à consulta* Q, $\mathscr{A}^Q$:

$$\mathscr{A}^Q = \sum_{i \in Q}(d_i) + R_Q \tag{2.6}$$

ou seja, *resposta exata + ruído aleatório.*

EXEMPLO PRÁTICO:
Considere o banco $d = (0, 0, 1, 0, 1, 1, 0, 1)$ e a função $\mathcal{E}(n) = n$. Deseja-se fazer a consulta $Q = \{1, 3, 5\}$ ao banco d:

1. Obtêm-se a resposta exata $\sum_{i \in Q} d_i = 1$
2. Ruído aleatório R_Q é gerado como $R_Q = 2$. [a]
3. Banco protegido devolve a resposta pública $\mathscr{A}_Q = 3$

[a] Poderia ser qualquer número desde que $|R_Q| < \mathcal{E}(8) = 8$, incluindo números negativos

Note que a adição de ruído aleatório pode arruinar completamente a pesquisa que se deseja fazer no banco de dados, criando uma resposta longe demais da realidade. Por este lado, seria interessante portanto que a função $\mathcal{E}$ fosse a menor possível. Entretanto, uma função baixa demais comprometeria a privacidade do banco. Calibrá-la para garantir a usabilidade dos dados e a privacidade não é uma tarefa fácil.

Por último, denotaremos por $neg(n)$ uma função que é assintoticamente menor que qualquer inversa de polinomial.[3] Ou seja,

$$neg(n) < \frac{1}{n^c} \quad \forall c > 1 \tag{2.7}$$

O intuito do uso da notação neg é de demonstrar que uma função que seja menor que $neg(n)$ tem **valor ínfimo**, ou seja, tem um valor tão pequeno que pode praticamente ser desconsiderado. Servirá para indicar eventos probabilísticos que possuem baixíssima chance de ocorrência.

2.2.1 Métodos de garantia de privacidade

2.2.1.1 Auditoria de Consultas

A Auditoria de consultas é uma técnica que tem como objetivo proteger a privacidade do banco de dados[2]. As consultas são submetidas à análise (em geral não humana) antes de serem respondidas. Consultas que sejam consideradas comprometedoras demais podem não ser respondidas com objetivo de impedir que quem esteja consultando obtenha dados comprometedores.

O primeiro problema da auditoria de consultas é que, comprovadamente, a auditoria de consultas é um problema NP-Completo[6]. Verificar se uma consulta e sua resposta possui potencial para violar a privacidade não é uma tarefa computacionalmente realizável. Além disso, a comparação de informações obtidas a partir de consultas respondidas pelo auditor

aliadas às consultas negadas já pode levar a um vazamento parcial ou total dos dados do banco[3].

2.2.1.2 Perturbação do Banco de Dados

A perturbação de banco de dados é uma técnica em que as consultas não são realizadas sobre o banco original, e sim em uma versão perturbada, em que os dados não correspondam 100% aos reais. Alguns de seus atributos, por exemplo, podem ser substituídos por outros que sigam a mesma distribuição. Assim, ainda se pode obter dados preciosos sobre a natureza das informações (como distruibuição e dependência com outras variáveis) mas não se pode revelar individualmente a origem. Esta técnica é chamada de *swapping*[8].

Uma variante da perturbação do banco de dados é a perturbação das consultas. Nesta modalidade, adiciona-se um ruído aleatório sobre as respostas de cada consulta requisitada ao banco de dados. Esta modalidade será a explorada no projeto e no simulador, consequentemente a melhor descrita no decorrer do texto, servindo como base para as discussões. Mais especificamente, o objetivo é a análise do limite inferior que o ruído precisa ter para que garanta que alguém mal-intencionado não possa romper a privacidade dos dados utilizando sucessivas consultas em um algoritmo polinomial.

2.3 Reconstrução da base de dados

Introduzidos os conceitos e motivações iniciais, estamos prontos para prosseguir para o real problema que queremos desenvolver. Queremos analisar as possibilidades de reconstrução do banco de dados d. Isto é, dado um banco de dados de tamanho n (que não conhecemos por completo) e m respostas com ruído de consultas quaisquer, determinar o banco d. Mais importante que isso, responder o questionamento: **Qual deve ser a magnitude de $\mathcal{E}$ para que a privacidade não possa ser violada?**

Para tentar fazer a reconstrução, assumimos o modelo de perturbação sobre as consultas (2.2.1.2), e que não existe limitação quanto ao número de consultas a base. Além disso, o tempo que o banco leva para responder uma consulta é $O(1)$.

Dado um número $\epsilon \in]0,1[$, queremos contruir uma sequência $y \in \{0,1\}^n$ tal que y esteja afastado de d em no máximo ϵn. Definimos afastamento com a seguinte função:

$$dist(y,d) = |\{i : y_i \neq d_i\}| \tag{2.8}$$

ou seja, a **quantidade de posições em que y e d diferem**, também conhecido como **distância de Hamming[5].**

2.4 Adversário Exponencial

A primeira possibilidade de reconstrução do banco de dados é bem óbvia: o ataque exponencial. Gerar todas as consultas possíveis (portanto 2^n consultas) e obter suas respostas. Depois, testar sequencialmente as possibilidades de banco de dados (mais 2^n possibilidades) e verificar se encaixam nas respostas obtidas pelas consultas. Note que a verificação de encaixe deve levar em conta a perturbação das respostas públicas descritas em 2.2.

Seja $[n]$ todos os subconjuntos de $1, 2, 3, ..., n$. O algoritmo de ataque exponencial ao banco pode ser descrito então como:

1. Obter todas as consultas possíveis $Q^t = \{Q : Q \subseteq [n]\}$
2. Obter todas as repostas públicas das consultas $\mathscr{A}^t = \{\mathscr{A}^Q : Q \in Q^t\}$
3. Obter todos os bancos possíveis $c \in \{0,1\}^n$. Para cada um, verificar se:

$$|\sum_{i \in Q} c_i - \mathscr{A}^Q| \leq \mathcal{E}(n), \forall Q \in Q^t$$

. Caso a verificação seja verdadeira, o banco foi encontrado. Retornar c.

É fácil ver que o algoritmo acima funciona, já que ele testa todas as possibilidades existentes na formação dos dados. Em especial, a base de dados original com certeza é uma delas.

Note que esta discussão não informa nada sobre limites (inferiores ou superiores) a função $\mathcal{E}(n)$. Isso porque, neste caso, realmente **não importa qual a magnitude da perturbação, o método de força bruta consegue deduzir a resposta.** A aplicação deste algoritmo no mundo real, entretanto, não é possível, por ser $O(2^n)$. Um banco de dados de 170 posições (um número bem pequeno considerando o tamanho de alguns bancos de dados) levaria mais iterações para ser resolvido que a quantidade estimada de átomos do planeta Terra.

Para ser aplicável, o método deve ser tal que a sua execução leve tempo polinomial.

2.5 Adversário Polinomial

Reconstruir o banco de dados utilizando um algoritmo polinomial é uma tarefa um pouco mais difícil. Em especial, provar que o método funciona, e qual o seu limite de atuação.

No caso de consultas polinomiais, claramente não se pode testar todas as possibilidades. A abordagem ideal passa por restringir ao máximo a cada consulta e resposta obtida os reais valores do banco de dados. E essa restrição é mais abrangente quanto menos restritiva for a função $\mathcal{E}$. Analisar o quão restritivas precisam ser as respostas para que um adversário

polinomial restrinja o espaço do banco de dados o suficiente e quebre a privacidade é tarefa importante para a proteção dos dados.

No caso, demonstraremos pelo lado adverso: mostrando um algoritmo que em tempo polinomial encontra o banco de dados original com base em consultas aleatórias, com chance de erro ínfima, desde que $\mathcal{E}(n) = o(\sqrt{n})$, provando assim que qualquer banco que não possua ao menos uma perturbação de ordem $o(\sqrt{n})$ está vulnerável a decriptação. O algoritmo foi retirado do trabalho de [3], embora a demonstração tenha sido alterada para melhor ilustrar seus efeitos ao leitor deste trabalho.

Antes de começar, vale a ressalva de que o algoritmo usa consultas aleatórias com objetivo de levar a demonstração e os resultados ao campo da estatística, tornando possível uma análise sobre os limites de erros e probabilidade de sucesso. O algoritmo aqui utilizado poderia ter uma aproximação mais determinística, mas a demonstração de seus resultados seria traçado por outro caminho.

1. Obter $t = nlog^2(n)$, o número de consultas aleatórias que serão realizadas.
2. Obter o conjunto $Q^t \subset [n]$, contendo t consultas escolhidas ao acaso.
3. Obter o conjunto $\mathscr{A}^t = \{\mathscr{A}^Q : Q \in Q^t\}$, as respostas públicas das consultas.
4. Seja $c_1, c_2, ..., c_n$ as variáveis que representam cada posição do banco de dados originais

   Resolver o problema linear:

   Max $c_1 + c_2 + c_3 + ... + c_n$

   Sujeito a: $\mathscr{A}^Q - \mathcal{E} \leq \sum_{i \in Q} c_i \leq \mathscr{A}^Q - \mathcal{E}, \quad \forall Q \in Q^t$

   $0 \leq c_i \leq 1, \qquad para \quad 1 \leq i \leq n$
5. Arredondar cada c_i para 1 se $c_i > \frac{1}{2}, \quad 0$ caso contrário

Antes de proceder a demonstração, alguns comentários devem ser feitos.

Em primeiro lugar, a prova não inclui absolutamente nada sobre a função objetivo do problema linear. Em realidade, de fato, poderia ser qualquer outra. As restrições já limitam o espaço de soluções possíveis o suficiente para aceitar somente uma resposta, independentemente da função objetivo.

Além disso, é fácil notar também que o problema linear sempre possui uma solução, já que o próprio banco de dados original d está com certeza inserido no espaço aceito pelas restrições. Resta então demonstrar que será de fato essa resposta obtida pela resolução do problema:

2.5.1 Demonstração do Algoritmo polinomial

A demonstração segue pelo caminho de que, caso um candidato ao banco de dados se afaste do banco original, existe uma alta probabilidade de que uma das $nlog^2(n)$ deve desclassificá-lo.

Fixemos um número $\varepsilon > 0$, e com ele, um vetor $x = (x_1, x_2, ..., x_n) \in [0,1]^n$ de maneira que:

$$|\{i : |x_i - d_i| > \frac{1}{3}\}| > \varepsilon n \tag{2.9}$$

ou seja, de maneira que a quantidade de componentes de x que se distanciam de d em mais de $\frac{1}{3}$ seja ao menos εn. Apelidaremos este x ao longo deste capítulo por *"x ruim"*, por estar demasiadamente afastado do objetivo d.

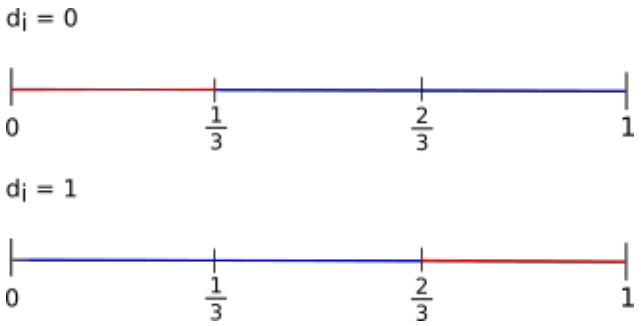

Figura 1 – Os componentes do vetor x escolhidos para estarem distantes de d devem atingir as exigências conforme os intervalos da figura: $x_i \in]\frac{1}{3}, 1]$ caso $d_i = 0$, e $x_i \in]\frac{1}{3}, 1]$ caso $d_i = 1$

A prova segue do seguinte lema:

Lema 1: *Dada uma consulta Q escolhida ao acaso, a probabilidade de que x esteja contido no intervalo $[\mathscr{A}^Q - \mathcal{E}, \mathscr{A}^Q + \mathcal{E}]$ é de no mínimo δ, onde $\delta > 0$, desde que $\mathcal{E} = o(\sqrt{n})$.[3]* [1]

Note que o intervalo é justamente aquele que serve de restrição na formação das restrições do problema linear do algoritmo.

Do lema, pode-se adotar a interpretação inversa: que a probabilidade de x estar de fato, contido no intervalo é de no máximo $1 - \delta$, onde $1 - \delta < 1$. Conforme aumenta-se o número de consultas, portanto, a chance deste mesmo x estar no intervalo aceito por todas elas (e portanto, na região aceitável do problema linear) é de $(1 - \delta)^t$

[1] O Lema decorre do teorema da inegualdade de Azuma[1]. O teorema tem como objetivo verificar limites para ocorrência de eventos estatísticos, e sua demonstração foge do escopo deste trabalho. O leitor pode encontrar a demonstração nas referências.

Com o intuito de chegar a uma limitação mais estreita de probabilidade, vamos discretizar o espaço possível para x. Seja

$$K = \{0, \frac{1}{k}, \frac{2}{k}, \frac{3}{k}, ..., \frac{k-1}{k}, 1\} \tag{2.10}$$

a discretização em k pedaços equidistantes do intervalo $[0, 1]$, e denotando por $\chi^n(k)$ a grade que distretiza em k pedaços o espaço $[0, 1]^n$. A discretização torna o número de *"x ruim"* na grade finito. Seja

$$L_\varepsilon^n \tag{2.11}$$

o conjunto de *"x ruim"* na grade $\chi^n(k)$

Dada essa discretização, consideremos o evento:

> EVENTO E: Dado o conjunto Q^t de t consultas aleatórias, existir um ponto $x \in L_\varepsilon^n$ tal que x está em todos intervalos $[\mathscr{A}^Q - \mathcal{E}, \mathscr{A}^Q + \mathcal{E}], \quad \forall Q \in Q^t$ gerados pelas consultas.

Observe que o conjunto L_ε^n **não é aleatório**, diferentemente das t consultas.

Chegar ao valor exato da probabilidade do evento não é uma tarefa simples. Principalmente porque passa por um problema de contagem: determinar quantos elementos existem em L_ε^n dado um certo banco d.[2]

Vamos então estimar a probabilidade tentando encontrar um limite superior: O conjunto L_ε^n está inteiramente contido no conjunto $\chi^n(k)$. Logo, o número de elementos da intersecção de ambos com certeza é menor que o número de elementos em $\chi^n(k)$, facilmente calculável $((k+1)^n)$.

A probabilidade do evento E então possui como limite superior o número :

$$(k+1)^n(1-\delta)^t \tag{2.12}$$

Note que, para qualquer t que cresce mais rápido que n, a expressão 2.12 é $neg(n)$. Logo, **a probabilidade que um *"x ruim"* não seja desclassificado pelas consultas é ínfimo**.

Por fim, considere o arredondamento $\overline{c}$, obtido aproximando o resultado do algoritmo para o componente mais próximo de $\chi^n(k)$. Considere também a hipótese que $\overline{c} \notin \chi^n(k)$. Então, de maneira contrária a definição 2.9, temos que:

$$|\{i : |x_i - d_i| > \frac{1}{3}\}| \leq \varepsilon n \tag{2.13}$$

[2] Uma tentativa de aproximar este número será descrita mais para frente

Note que, para um componente único de $\overline{c}$, $|\overline{c_i} - d_i| \leq \frac{1}{3} \Rightarrow c_i = d_i$:

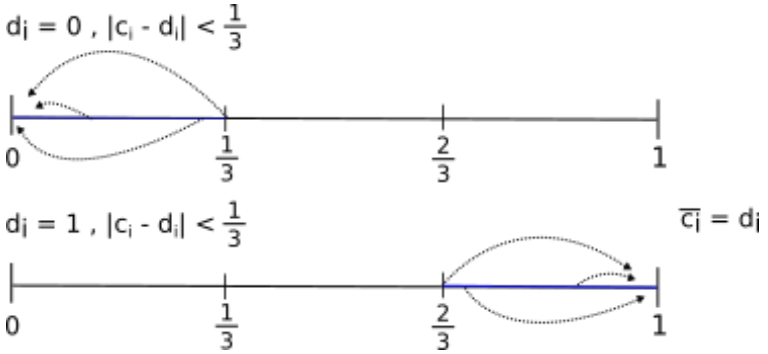

Figura 2 – Note também que é possível imposição de uma distância máxima mais branda que $\frac{1}{3}$. Em realidade, qualquer distância estritamente menor que $\frac{1}{2}$ segura o argumento.

Logo,a partir de 2.13 podemos reescrever da seguinte forma:

$$|\{i : |c_i - d_i| \neq 0\}| \leq \varepsilon n \tag{2.14}$$

Portanto, temos que $\overline{c} \notin \chi^n(k) \Rightarrow dist(c, d) \leq \varepsilon n$, concluindo por fim a demonstração.

A demonstração nos permite concluir que, com alta probabilidade $(1 - neg(n))$ um *"x ruim"* não passará pelas restrições que as t consultas nos fornecem. Logo, com a mesma probabilidade, $\overline{c} \notin \chi^n(k)$, e por fim, $dist(c, d) \leq \varepsilon n$

3 Simulações

Para testar algumas hipóteses e aferir o funcionamento empírico dos dois algoritmos apresentados (exponencial 2.4 e polinomial 2.5), foi desenvolvido um simulador. Criado em linguagem Python, quatro versões diferentes foram criadas, cada uma com intuitos diferentes.

Todos os testes foram executados em um computador com processador de 2 núcleos de 2.3GHz e 4Gb de RAM.

3.1 Simulação do algoritmo exponencial

Uma versão do algoritmo explicado em 2.4 foi criada. A assertividade do algoritmo está sobre qualquer tipo de suspeita. Isso porque busca todas as possibilidades de banco de dados exaustivamente, até encontrar o que satisfaça todas as condições.

Esta simulação tem como principal objetivo verificar até onde é possível executá-lo. Por ser um algoritmo $O(2^n)$, pode-se imaginar que estoure o tempo ou a memória rapidamente conforme cresce a dimensão n do banco de dados.

Os testes sucederam na tentativa de quebrar a segurança de bancos de até 16 posições, conseguindo chegar ao banco real em 100% dos casos. Bancos maiores não podem ser decriptados. Mesmo quando se utilizando de técnicas que tornam o gasto de memória constante ($O(1)$), o gasto de tempo torna sua utilização inviável.

n	Tempo de execução (em segundos)
10	0,01
13	3,09
15	34,59
16	65
18	529,09

Tabela 1 – Tempo de execução para o algoritmo exponencial de decriptação do banco de dados.

3.2 Simulação do algoritmo polinomial

A versão que simula a execução do algoritmo polinomial para decriptação do banco de dados (explicada em 2.5) foi testada para quebrar a segurança de bancos de dados de tamanho até $n = 500$. Para bancos maiores que isso, o computador em que os testes foram executados

não apresenta memória suficiente para aguentar a simulação e falha. Nada indica, entretanto, que o uso de mais poder computacional não fosse suficiente para conseguir decriptar bancos maiores com a mesma facilidade que os menores.

Quanto ao tempo de execução, para a resolução do problema linear descrito em 2.5, foi utilizado o algoritmo Simplex. Todos os testes foram executados duas vezes, utilizando como função objetivo a maximização da soma dos elementos do banco de dados, seguido então pela minimização. Um teste só é considerado sucedido se ambas as resoluções foram capazes de determinar corretamente o banco de dados original por meio das consultas aleatórias.

n	Tempo de execução (em segundos)
10	$< 0,01$
50	0,03
100	0.78
200	6.83
300	55.28
400	167.63
500	331,43

Tabela 2 – Tempo de execução para o algoritmo polinomial de decriptação do banco de dados. Para permitir uma comparação coerente, foi utilizado o tempo de execução do algoritmo para minimização da soma do banco.

3.3 Contagem de L_ε^n

Relembrando a seção 2.5.1, o conjunto L_ε^n é definido como *o conjunto de "x ruim"na grade de discretização* $\chi^n(k)$, onde "x ruim"é definido como um vetor de $[0,1]^n$ tal que ao menos εn componentes se afastem do banco original d em $\frac{1}{3}$. Toda a demonstração do capítulo anterior baseia-se na probabilidade de que os elementos deste conjunto são descartados pelas consultas estatísticas ao banco de dados original (e portanto, o evento E não ocorra).

Como maneira auxiliar de mostrar que, de fato, a probabilidade de ocorrência do evento E é muito pequena, um teste de Monte-Carlo foi executado. Cada um dos testes se trata da geração de um banco de dados aleatório, $n(log^2(n))$ consultas e suas respectivas respostas públicas (conforme definido em 2.6). Depois, uma massa numerosa de pontos na grade de discretização $\chi^n(k)$ escolhidos ao acaso, e verificado se:

a O ponto gerado está ou não em L_ε^n

b O ponto gerado está ou não na região aceitável do problema linear definido em 2.5.

Os testes foram executados por três dias, utilizando bancos de dados de tamanho $n = 8$ e $n = 5$. Note que o tamanho do banco é bem pequeno, gerando menos pontos na grade, menos restrições ao problema, e intuitivamente, uma chance maior de aceitação dos pontos.

n	Número de pontos gerados	a	b	a & b
5	596.540.912	594.206.989	32.706.020	2.283.410
8	296.305.303	296.263.261	0	0

Tabela 3 – O número gerado pelos testes indica que a chance de que um *"x ruim"* esteja na região aceitável decresce rapidamente conforme o tamanho do banco aumenta. Para n=5, os testes de MonteCarlo indicam uma probabilidade de **0,38%**, enquanto para n=8, **nenhum caso foi detectado**. Os testes para n maiores tomam muito tempo, inviabilizando a geração de consultas o suficiente para criar um teste significativo.

4 Conclusão

Este projeto condensa os conceitos essenciais desta temática e que servem como fundação de uma série de outros temas como: privacidade individual nas arestas e nos vértices de um grafo, k-anonimização de um banco de dados, entre outros. Em especial, da obtenção de geradores de grafos que possam proteger a privacidade diferencial de seus nós, os *Graphons*, que foi o estopim no interesse pelo tema.

Os conceitos desenvolvidos e os algoritmos testados demonstram que um banco de dados estatístico que seja perturbado em até $o(\sqrt{n})$ pode ser facilmente decriptado por um adversário polinomial. Possivelmente, a maioria destes bancos que hoje estão disponíveis a utilização pública (seja com a adição de ruído ou mesmo simples remoção da identificação dos registros individuais) não possui qualquer garantia de privacidade. O uso de sucessivas consultas, e o agrupamento do conhecimento que é extraído de cada uma delas, quebra sua proteção. Os resultados das demonstrações são ratificadas pelas simulações, que demonstram que a probabilidade de falha da decriptação decresce rapidamente quanto maior o banco de dados.

Além disso, a camada de proteção que os ruídos aleatórios impõem geram como efeito colateral um distanciamento da base real. Note que esse distanciamento gerado já pode, potencialmente, privar os dados de um uso realmente útil (para a maioria das aplicações $o(\sqrt{n})$ é um preço alto de mais a se pagar). Por conseguinte, a tática mais óbvia para aumentar a proteção, adicionar mais ruído, é problemática por gerar mais distanciamento, comprometendo ainda mais a usabilidade.

Embora academicamente o uso de ruído aleatório seja o ponto de partida de qualquer tentativa de conservação da privacidade, seu uso em dados reais é limitado demais para ser factível.

Referências

[1] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004.

[2] Francis Y Chin and Gultekin Ozsoyoglu. Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, (6):574–582, 1982.

[3] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. pages 202–210, 2003.

[4] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. pages 265–284, 2006.

[5] Richard W Hamming. Error detecting and error correcting codes. *Bell Labs Technical Journal*, 29(2):147–160, 1950.

[6] Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. Auditing boolean attributes. *Journal of Computer and System Sciences*, 66(1):244–253, 2003.

[7] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE, 2009.

[8] Steven P Reiss. Practical data-swapping: The first steps. *ACM Transactions on Database Systems (TODS)*, 9(1):20–37, 1984.

[9] Arie Shoshani. Statistical databases: Characteristics, problems, and some solutions. In *VLDB*, volume 82, pages 208–222, 1982.