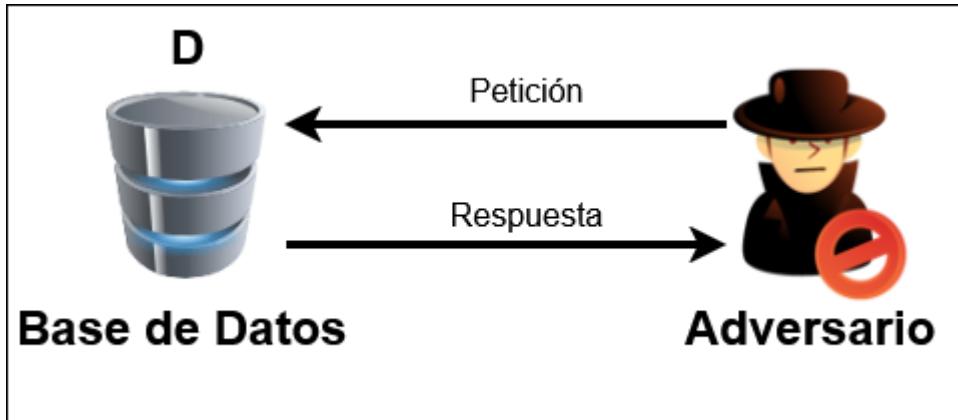
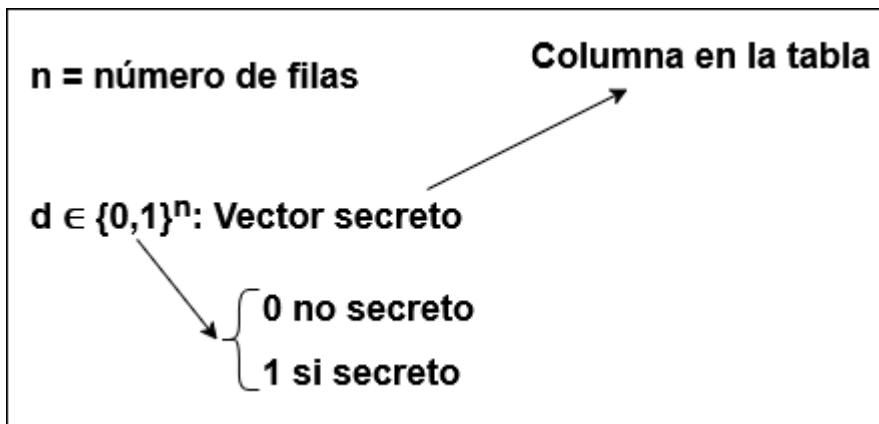


Ataques de reconstrucción de Bases de Datos

Un adversario quiere inferir algo de una base de datos mediante peticiones. Un ataque de inferencia está pensado para reconstruir una base de datos previamente curada. Se recomienda curar la base de datos evitando que esta tenga datos demasiado detallados.



Queremos proteger la base de datos contra el exceso de inferencias. El adversario puede tener una estrategia de peticiones dinámicas que van cambiando en función a la información que van obteniendo. Para proteger los datos, se pueden denegar respuestas en función a lo peligrosas que puedan ser con respecto a la privacidad. El problema de denegar información es que a su vez puede dar información sin querer al adversario. Por ejemplo, si se pide el mayor sueldo de un rango de empleados, sale X cifra, si luego hacemos la misma petición quitando un empleado y aparece una cifra diferente, podemos inferir el sueldo de dicho empleado.



El atacante hace preguntas usando un vector que tiene los datos tratando de descubrir si un dato es secreto o no.

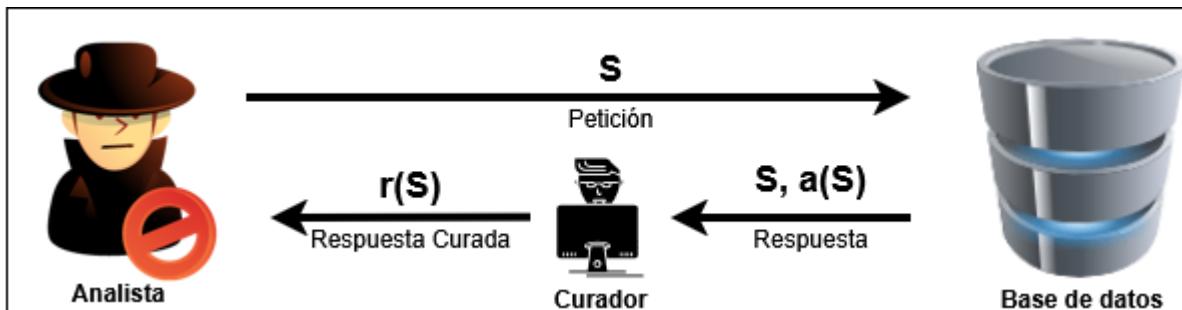
Censo Ficticio de EEUU

Se construyó una base de datos falsa simulando el censo de EEUU. Algunos datos como la edad han sido suprimidos para ciertas personas con el objetivo de proteger contra ataques de inferencia al haber demasiada poca gente con datos que coincidan con ellos. A pesar de estar estos datos

eliminados, se da acceso a datos estadísticos como la media y la mediana, lo que permite ir induciendo poco a poco las edades que han sido ocultadas. Con todo esto, se puede proponer un sistema de inecuaciones que se puede aplicar a un algoritmo solver, resultando en que se pueden obtener los datos ocultos en dicha Base de Datos. Como resultado, el censo hizo un ataque de reconstrucción a su base de datos en 2010 usando estos mismos principios, resultando en la reconstrucción de un 46% de la base de datos sin errores y un 71% de la base de datos tolerando algunos errores. En total se pudo identificar a 50 millones de personas. A raíz de esto el censo comenzó a usar privacidad diferencial en 2020, añadiendo ruido a las respuestas para dificultar ataques de reconstrucción.

Curando Respuestas

Responder a una petición S con la respuesta verdadera $a(S)$ viola la privacidad, si el adversario quiere descubrir un dato oculto para cierta fila, simplemente tiene que construir un vector de peticiones con un 1 en la fila y 0 en el resto. Para evitar violar la privacidad la respuesta $r(S)$ debe ser una versión de $a(S)$ con ruido añadido. Para preservar la utilidad de los datos se deben establecer unos límites de distorsión como, por ejemplo, $\|r(s) - a(s)\| \leq E$



Ataque de reconstrucción lineal

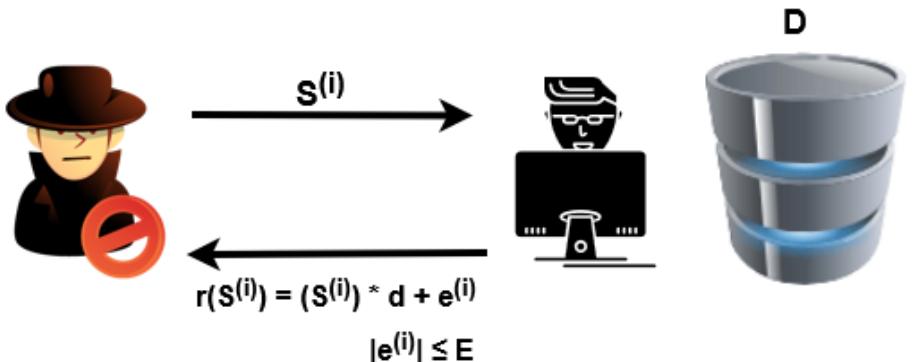
Teorema: Si el analista puede realizar 2^n peticiones y el curador añade ruido E , entonces el adversario puede reconstruir la base de datos entera salvo por $4E$ posiciones.

Fase 1

- El analista envía todas las peticiones posibles a la base de datos y almacena las respuestas.

Lista de posibles peticiones

$S^{(1)} = [0, \dots, 0, 1]^T$
$S^{(2)} = [0, \dots, 1, 0]^T$
$S^{(3)} = [0, \dots, 1, 1]^T$
...
$S^{(2^n)} = [1, \dots, 1, 1]^T$



```
for i = 1:2^n
    Analista envía s(i)
    Curador responde con r(s(i)) ) s(i)^T * d + e(i)
```

Fase 2

- El analista busca candidatos válidos

Lista de Candidatos

$C^{(1)} = [0, \dots, 0, 1]^T$
$C^{(2)} = [0, \dots, 1, 0]^T$
$C^{(3)} = [0, \dots, 1, 1]^T$
...
$C^{(2^n)} = [1, \dots, 1, 1]^T$



```
for k = 1:2^n
    candidate = 'valid'
    for i = 1:2^n
        if |r(s(i)) - (s(i))^T * c(k)| > E:
            candidate = 'not valid'
            break
    if candidate = 'valid' : store
    c(k) in lista de candidatos válidos
```

Resumen de ataque de reconstrucción lineal

A menos que haya un límite de peticiones a la base de datos, es posible crear una reconstrucción casi perfecta de $4E$ entradas de la base de datos. Por ejemplo, si $E = n/401$, la base de datos puede ser reconstruida para al menos el 99% de las entradas (E es el error máximo que se introduce en las respuestas, por lo que no debe ser demasiado grande.). Sacrificando algo de la precisión, es posible reconstruir estadísticamente el vector secreto con muchas menos peticiones.

Ataque de Reconstrucción Lineal Probabilístico

Dinur y Nissim propusieron un ataque que funciona en $O(n)$ peticiones cuando el curador añade un ruido de $E=O(\sqrt{n})$, permitiendo reconstruir toda la base salvo $O(\sqrt{2})$ posiciones usando programación lineal. Para evaluar la magnitud del sonido, se toma una muestra de una población que satisface una propiedad con probabilidad p y tome n muestras, la media es np y la desviación típica es $\sqrt{p(1-p)} \sqrt{n}$, por lo que R es requerido para evitar que este tipo de ataque sea mayor que el error de las muestras.

Desafío Aircloack's Diffix

La compañía Aircloack, que vendía el producto Diffix, prometía privacidad en base a curar las respuestas de las bases de datos. Esta compañía lanzó un concurso con un premio de 5000\$ para ver si su producto respondía a los ataques de reconstrucción como se esperaba. La cantidad de ruido que se añadía en cada respuesta era la raíz cuadrada del número de condiciones. Estaban prohibidas las operaciones de tipo OR en la base de datos y, en caso de que la salida tuviera pocas entradas, la base de datos no respondería. Se planteó una forma de interrogar la base de datos de forma que las peticiones cubrieran muchos elementos de la base de datos de una forma más o menos aleatoria y que tuvieran un número de condiciones muy pequeño. Como resultado se obtuvo la siguiente consulta SQL:

```
SELECT COUNT(clientId)
FROM loans
WHERE FLOOR(100*((clientId*2)^0.7)+0.5) = FLOOR(100*(clientId*2)^0.7)
AND clientId BETWEEN 2000 AND 3000
AND loanStatus = 'C'
```

Con esta consulta se pudo reconstruir la base de datos a pesar de las medidas tomadas.

From:
<http://www.knoppia.net/> - Knoppia



Permanent link:
http://www.knoppia.net/doku.php?id=pan:ataques_recontruccion_v2

Last update: 2026/01/04 15:56